A VOTING-BASED IMAGE SEGMENTATION SYSTEM

Valentin Gabriel MITREA [1*]
Mihai-Cristian PÎRVU [2]
Mihai-Lucian VONCILĂ [3]
Costin-Anton BOIANGIU [4]

ABSTRACT

Image segmentation is an important topic in the field of Computer Vision and has numerous practical applications. It is often used as a preprocessing step for other higher level image processing algorithms such as: text analysis, object identification, feature extraction, etc. However, there is no image segmentation technique that can produce perfect results on any type of image. Numerous algorithms exist and each has its upsides and downsides depending on the input. This paper proposes two voting algorithms that combine the results of some well-known segmentation techniques into a final output which aims to be, in as many cases as possible, better than the individual segmentations.

KEYWORDS: *image processing, segmentation, voting technique, region growing, cluster, graph, mean shift*

1. INTRODUCTION

Image segmentation represents a class of image processing algorithms that have the purpose of organizing an input image into groups of pixels. These groups are formed based on some criterion such as color intensity. Therefore, all the pixels from a particular segment have to be similar to each other.

Considering the fact that no image segmentation technique can provide ideal results on any given image, we can split the segmentation algorithms into two categories: the ones that have the tendency to perform oversegmentation and the ones that have the tendency to perform undersegmentation.

Oversegmentation means dividing the image into a very large number of segments. This has the advantage of focusing on details, but the end result will be affected by the noise present in the image.

[1*] corresponding author, Engineer, "Politehnica" University of Bucharest, Bucharest, Romania, mitrea.valentin@gmail.com
[2] Engineer, "Politehnica" University of Bucharest, Bucharest, Romania, mihaicristianpirvu@gmail.com
[3] Engineer, "Politehnica" University of Bucharest, 060042 Bucharest, Romania, mihai.voncila@gmail.com
[4] Professor PhD Eng., “Politehnica” University of Bucharest, Bucharest, Romania, costin.boiangiu@cs.pub.ro

On the other hand, undersegmentation means dividing the image into a small number of segments. This has the advantage of removing noise at the cost of ignoring some details from the image.

In figure 1, it can be seen that the popular Lena image segmented two times. On the second row, there is an oversegmented result (to the left) and an undersegmented result (to the right).

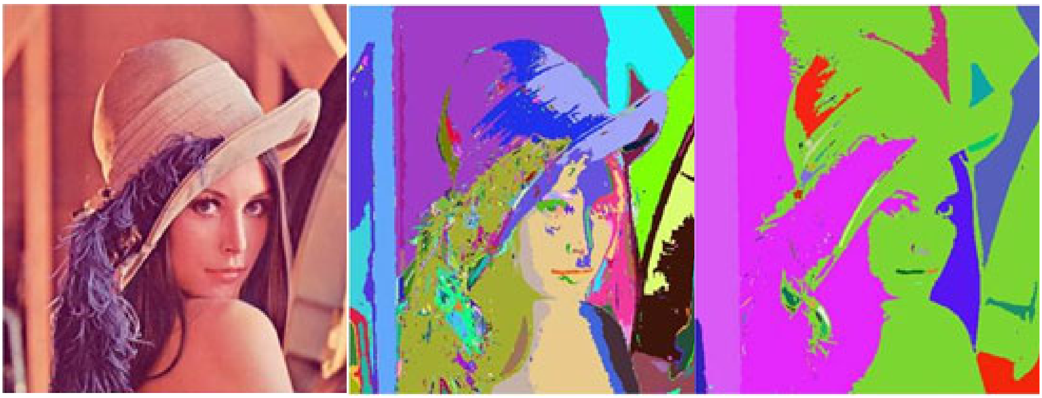


Figure 1. Comparison between oversegmentation and undersegmentation

The goal of this paper is to present a voting technique that combines the results of some well-known image segmentation algorithms. On their own, these algorithms do not provide the optimal results, but by combining them through the voting technique, it is expected to generate a final, improved output.

2. RELATED WORK

Ana Fred proposes in [1] a majority voting combination of clustering algorithms. The idea is that a set of partitions are obtained by running multiple clustering algorithms and, then, the outputs are merged into a single result. To achieve this, the results of the clustering algorithms are mapped into a matrix, called the co-association matrix, where each pair (i, j) represents the number of times the elements i and j were found in the same cluster.

And Fred's approach represented a starting point in [2] for the implementation of a voting image segmentation algorithm. In order to produce a meaningful result, the co-association matrix was used in combination with a weighted voting technique.

3. SEGMENTATION ALGORITHMS

In this section, there will be a discussion about the segmentation algorithms that generate the partial results which are used in our voting algorithms. The chosen algorithms for implementation: Region-Growing, Superpixel, Graph-Based and Mean-Shift. The selection was made in order to have a combination of oversegmentation and undersegmentation techniques. This way there are covered as many features from the input image as possible.

3.1. Region-growing

Region-Growing [3] is based on the fact that pixels which belong to the same region of an image must have similar properties. A region is created by starting from a single generator pixel and iteratively adding neighbors which verify a criterion. The criteria that was chosen for the implementation is that the color intensity between the neighbor pixel and the seed must be lower than a threshold value.

The algorithm starts from an input image where each pixel is labeled as not belonging to any region. While there are still pixels not allocated to a segment, the algorithm picks one and constructs a new region around it. Pixels are considered neighbors based on the four-neighboring connectivity rule. When the algorithm execution finishes, each pixel will be marked with an index that represents the region in which it belongs.

The main steps of the algorithm are the following:

```
• label = 0
• mark each pixel in the image as unlabeled
• while there are still unlabeled pixels
    o seed = one of the unlabeled pixels
    o Q = queue of pixels
    o labels[seed] = label
    o add seed to Q
    o while Q is not empty
        ▪ pixel = front of Q
        ▪ for each neighbor of pixel
            • if neighbor is unlabeled and neighbor is similar to seed
                o labels[neighbor] = label
                o add neighbor to Q
    o label = label + 1
```

A very important step of the algorithm is the selection of a new generator pixel. There are two implemented techniques for this choice: random and pseudo-intelligent. The first method will randomly pick a seed from the remaining unlabeled pixels, while the second will favor generator pixels that have a high number of similar neighbors and leave the others to the end.

3.2. Superpixel

The Superpixel algorithm [4] is a variant of the K-means clustering algorithm, where the input is an image and the output is a clustered image, with values for each position corresponding to the index of the cluster that position has been assigned to.

The K-means clustering algorithm is used in general to partition a dataset $\{x_1 \ldots x_n\}$ (each x_i is a D-dimensional point) into K clusters, where K is considered a hyper-parameter which is given. Each cluster j has a mean value μ_j (also called centroid) and each point x_i is governed by a 1-of-K vector $r_{i,j}$, where $r_{i,j} = 1$ means that x_i is in cluster j. The goal of the algorithm is to assign the points to a cluster such that the distance of any point x_i to its cluster mean μ_j is as small as possible. Formally, this can be interpreted as to minimize the following function, called the *distortion measure*:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{K} r_{i,j} * ||x_i - \mu_j||^2$$

This function is minimized using an iterative approach of 2 steps, called the Expectation and Maximization steps which are executed until a convergence criterion is achieved. Initially, the values μ_j are chosen at random, but this can be altered, to provide faster convergence and better results. In the Expectation step, each point is reassigned to the closest centroid based on a distance function:

$$r_i = argmin_j ||x_i - \mu_j||^2$$

Then, in the Maximization step, the centroids of each cluster are updated taking the mean value of the current points assigned to that cluster.

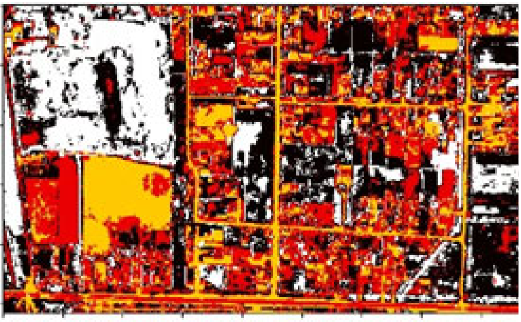
$$\mu_j = \frac{\sum_{i=1}^{N} r_{i,j} * x_i}{\sum_{i=1}^{N} r_{i,j}}$$

For the image segmentation problem, the K-means is used as follows. Each pixel $I_{i,j}$ represents a point x_i in the formulation above. The clustering is done in the color space, thus both the distance function and the centroids are represented in this space. Having a good initialization mechanism is one of the most important aspects of the K-means algorithm. A potential choice for initialization is the insurance that while the K-1 centroids are initialized at random, they are also initialized as far away as possible.

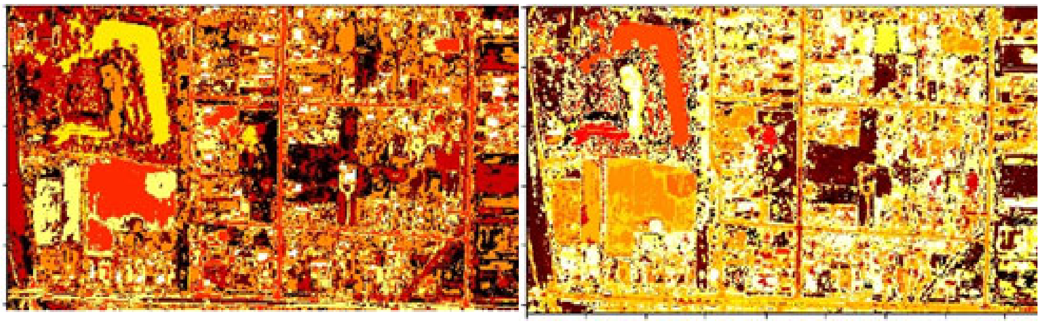
In practice, it is enough to reach a state where only a small percentage of the pixels change their centroids between two iterations, because reaching a convergent state might take too long and the improvement would be too slim to outweigh the time. A common value is 5% of the image.

Below is an example of the algorithm's output with different K values, on an image taken from satellite, basically showing how the algorithm can be used for undersegmentation and oversegmentation just by varying its only parameter.

Original imagine



Clustered imagine (K=4)



Clustered image (K=8)

Clustered image (K=20)

Figure 2. Superpixel segmented images

3.3. Graph-based

The algorithm chosen for implementation in this paper is a variation of the one proposed by Felzenszwalb and Huttenlocher [5]. The said algorithm works with two basic structures: a graph, that will contain the information relating the final segments of the processed image, and a disjoint set, used for optimizing the search inside the graph.

The Graph-Based technique will convert all the pixels of an image into nodes of a graph and place edges between the said nodes based on a four-neighboring connectivity rule. These edges represent the dissimilarity between two neighboring pixels. In order to obtain a segmented image as an end result, the algorithm will cut some edges based on a certain threshold. The algorithm also takes into account the fact that after segmenting the image some regions may end up being very small, thus it may be run multiple times to post-process the aforementioned small components.

The current implementation takes as an input the image we want to segment, the minimum size for each segmented region and the threshold at which to cut a certain edge in the graph and returns an image in which the colors have been changed based on the obtained regions.

The main steps of the algorithm are the following:

- *convert pixels from image to graph nodes*
- *create an edge between every two nodes based on a four-neighboring connectivity rule, with the weight of those edges being the distance between the colors of the two pixels*
- *create a disjoint set as follows*
    - *number of elements in set = number of nodes in graph*
    - *for each element in set*
        - *element pair = element*
        - *element threshold = threshold*
    - *for each edge in graph*
        - *element1 = find set element for node1 of the edge*
        - *element2 = find set element for node2 of the edge*

```
            ▪ if edge threshold <= element1 threshold and element2 threshold
              then
                  • combine element1 and element2 in set
                  • combined element threshold += edge weight
      o repeat to reduce the number of small regions
• create final image based on the elements in the set
```

Due to the fact that the algorithm has a tendency towards undersegmentation, running it once or twice is usually the best approach, as other runs after that will not guarantee reducing the number of small regions and will just result in a performance drop.

3.4. Mean-Shift

For Mean-Shift [6], the implementation found in the OpenCV library [7] was picked. The algorithm was selected to compensate for some of the inaccurate results the other algorithms might produce.

As a general idea, the algorithm works with two spaces, the image space (i.e. the space representing the distances between the image pixels) and the color space (i.e. the space representing the distance between the values of the image pixels). In order to create different regions from an input image, it fills the aforementioned spaces with windows that cover all the values, and then it tries to shift the said windows, taking into consideration the mean of the covered pixels, so that it will obtain the final segmented image. The algorithm stops when it reaches convergence across all windows.

The execution of Mean-Shift takes an input image to be segmented, the radiuses of both windows used for convergence, a maximum number of levels for the Gaussian pyramid, and offers a segmented image as output in which the pixel colors have been shifted. As it was mentioned, the algorithm tends to be slow sometimes, thus it is provided with two different termination criteria: convergence or a maximum number of iterations to run. This ends up favoring performance in detriment of stability. Due to the fact that the position of the windows is random, it might happen that running the algorithm multiple times will end up giving different results. However, the differences tend to be minimal if the values of the parameters are chosen correctly.

4. PARAMETER ESTIMATION

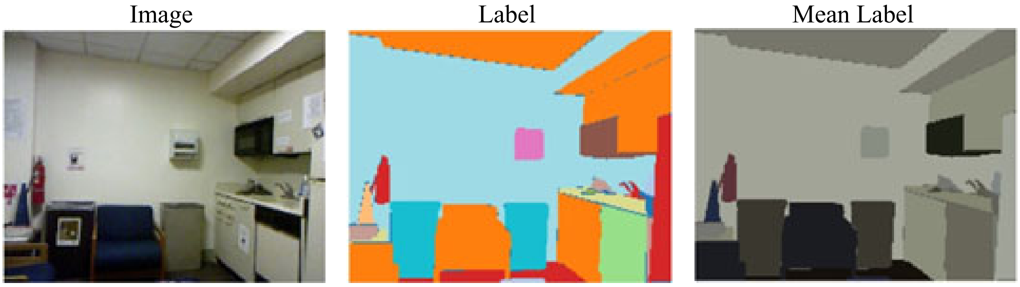
This chapter deals with the estimation of the parameters required by each algorithm described in section 3. The metric that is used to compute them will be described, then there will be a discussion about each algorithm independently and a verification of what values were found to work best and how they were picked.

In order to perform the estimations, it was used the NYU Depth dataset [8] which offers a large set of input images together with already segmented images. These labeled images serve as the ground truth in the calculations. The results from the segmentation and voting algorithms will be made to be as close as possible to the labels in the dataset.

The result of a segmentation algorithm on an image I is an image R_I that has the same shape as the initial image, but instead of having an intensity value for each position, it has an index of the region that position is part of. The problem that arises here is that if two different algorithms are run on the same image, the indexes of the same region will not correspond, and additional processing must be done. One approach is to create a similarity matrix as described here [1].

Thus, the labels from the dataset will be processed and the outputs of each algorithm in the following manner: these images, instead of having an indexed label for each position, will contain the mean intensity value of that region. Formally, this can be written as: $R_I(X) = M_{I,J}$ for all $X \in J$ and $M_{I,J} = \sum_{X\in J} \frac{I(X)}{|X|}$.

This computation can be seen in Figure 3, where there is an image from the dataset, its label and the calculated mean label. It can be observed that in the mean label each region looks roughly like the initial value, but it is still clustered. The algorithm can now work on this resulted image and apply its voting techniques on the intensity values directly, instead of working with arbitrary labels.



Image

Label

Mean Label

Figure 3. Mean Labeled image

Consider there is a Mean Segmented image from a segmentation algorithm A, called R_A, and a Mean Labeled image from the dataset, called R_L. Then, the comparison metric is defined as the Euclidean Distance between the two results: $L = \sqrt{\sum_{(i,j)}\left(R_L(i,j) - R_A(i,j)\right)^2}$. It should be noted that for a 640x480x3 image, as those in the NYU Depth dataset, the L value can vary from 0, where the result is identical to the label, to $\sqrt{640 * 480 * 3 * 255^2} = \sqrt{59{,}927{,}040{,}000} = 244{,}800$, when the label has only 0 values and the result has only 255 values (or the other way around). A second note that should be made here, is that, while the label does a good job at segmenting the image, it will usually compute the regions using a more semantic approach, rather than intensity based (which is how the algorithms used work). This can result in pretty big differences between the two results, but, nonetheless, the value L itself can be used to obtain the best parameters for the algorithms and to compare the algorithms with each other. Generally, a lower value means a better result and this is the metric we use when picking the parameters.

Now that the metric is defined, there must be a talk about what parameters are varied for each algorithm and see what values were chosen. For the Region-Growing algorithm, the threshold is varied and whether the initial seeds are chosen at random or using a more intelligent approach. For the Superpixel algorithm, there is only one parameter to vary, and that is the number of clusters it produces using the K-means technique. For Graph-

Based, there are two parameters: the minimum size and the denominator used in the standard deviation formula. Finally, for Mean-Shift, there are 3 parameters to find: the spatial and color radiuses and the maximum level of the Gaussian pyramid.

In order to obtain the ideal parameters, there were used 150 images, picked at random from the dataset, and applied the algorithms while computing the mean L value, as defined earlier, for each given set of parameters. Below are the results for each algorithm, and, as can be seen, even a small change can significantly improve the quality of the segmentation. In the end, there were kept only the best results, and were used in the voting scheme, which will be described in the next section.

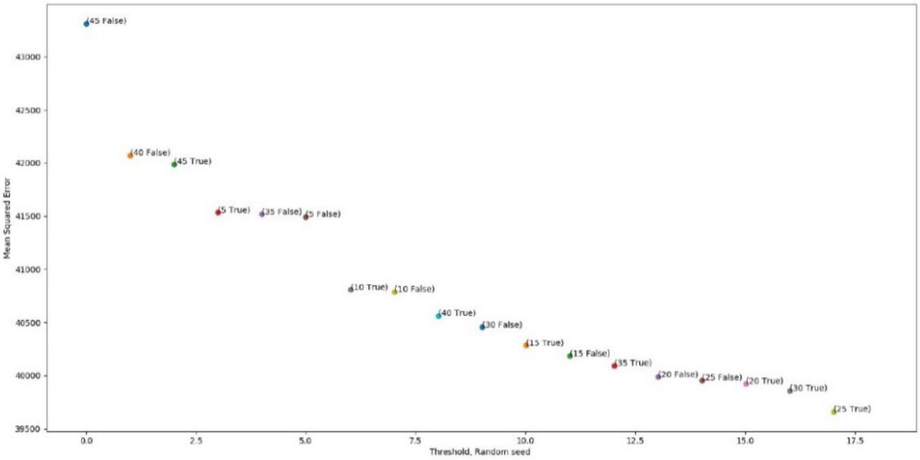


Figure 4. Region-Growing parameter choice results

For Region-Growing, the best results were using a threshold of 25 and random seed pixels.

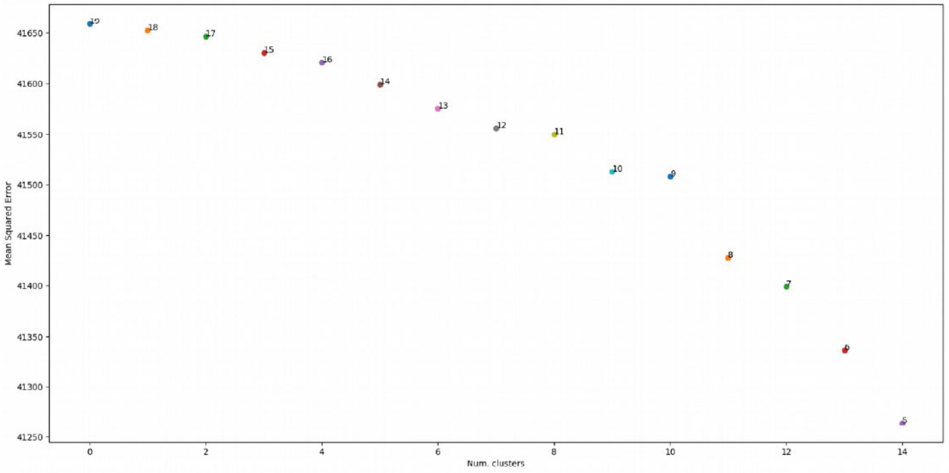


Figure 5. Superpixel parameter choice results

For Superpixel, as can be seen, a smaller amount of clusters yield the best results. Thus, there are used only 5 clusters as parameter for the algorithm.

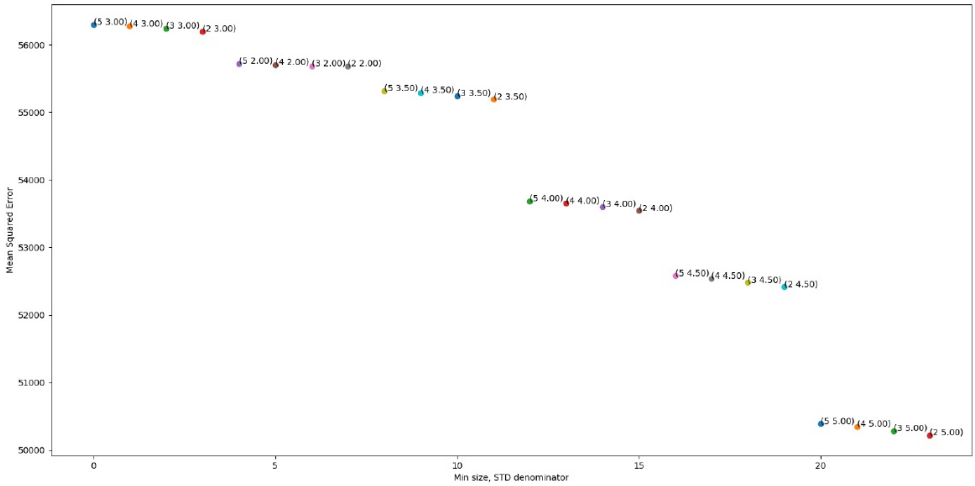


Figure 6. Graph-Based parameter choice results

For the Graph-Based algorithm, it can be seen that using a minimum size of 2 and a denominator of 5 gives the best segmentations.

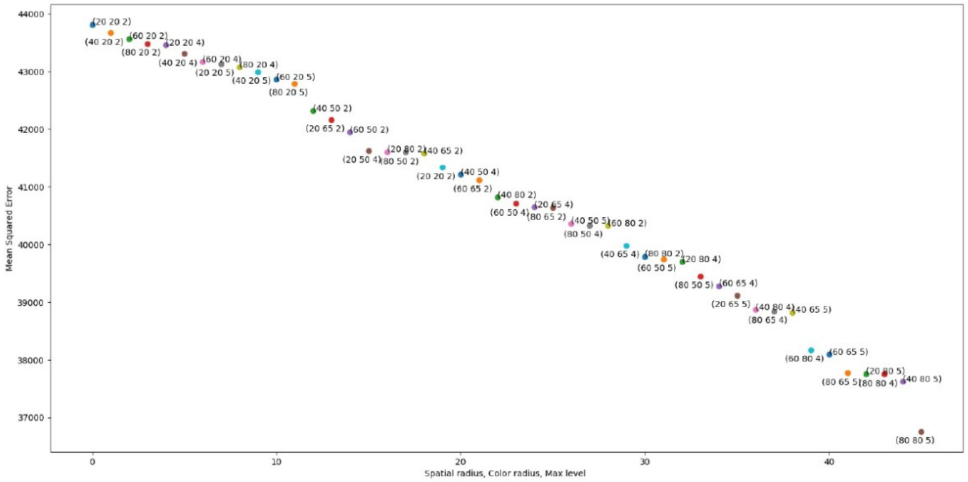


Figure 7. Mean-Shift parameter choice results

Finally, for Mean-Shift, the algorithm was able to get the values of 80 spatial radius, 80 color radius and a maximum level of 5. It should be noted that visually, these values, while having the lower L value, did not provide the best segmentation. So, a secondary set of parameters was selected: (40, 40, 5).

5. VOTING ALGORITHMS

A voting scheme is used to combine the results of the 4 algorithms described in chapter 3. This scheme takes the best from each algorithm, and tries to get a better segmentation in the end. In order to do this, there have been implemented two voting methods: Region Based (which is similar to the Region-Growing algorithm) and Weights Based (which uses a weighted sum scheme, followed by a re-clustering algorithm).

The algorithms are completely independent from each other, so, they are able to execute them in parallel. Also, all the 4 segmentation algorithms are run using the parameters estimated in chapter 4, thus the assumption that each algorithm produces the best possible individual result can be made. With this assumption, the combination of these partial results will lead to the best final result can be implied.

The voting methods used by us are executed on the Mean Segmented images computed by each segmentation technique. This allows the highlight on how applying the voting algorithm improves the result compared to each algorithm individually.

5.1. Region Based

The Region-Based technique takes the Mean Segmented output of each segmentation algorithm and tries to build regions using the information gathered from those partial results. The algorithm is based on the Region-Growing segmentation described in section 3.1 with a few differences.

The first one is that, in step a, when choosing the seed pixel, the algorithm simply takes the next unlabeled pixel from the image. For example, if pixels (0, 0) and (0, 1) are already labeled, then the generator pixel will be set as pixel (0, 2).

The second one is that, when deciding if a neighboring pixel should be added to the current region, the program does not look at the color intensities of the neighbor and seed pixel. Instead, it looks at the regions created by each of the algorithms and decide if the neighbor should be added to the current region.

The decision function will check if both Graph-Based and Mean-Shift say that the evaluated neighbor should be in the same region with the current position, and, if they both disagree, the two pixels will be in separate segments in the final image. This is because those two algorithms have the tendency to produce undersegmentation, and their capability of splitting segments should be trusted.

Furthermore, the decision function will check if both Region-Growing and Superpixel say that the evaluated neighbor should be in the same region with the current position, and, if they both agree, the two pixels will be in the same segment in the final image. This is because those two algorithms have the tendency to produce oversegmentation, so, if they say that two pixels are in the same region, then that is pretty likely to be correct.

5.2. Weights Based

The second voting method we used, is a bit different from the first one. The Mean Segmented image is also used, but in another way. It can be considered that each algorithm should contribute in a linear fashion, based on how well it performed during the parameter estimation. Thus, the final image will be a weighted sum of all algorithms:

$$V = w_R * R_R + w_S * R_S + w_G * R_G + w_M * R_M$$

In the above formula, the indexes R, S, G and M represent each of the 4 algorithms: Region-Growing, Superpixel, Graph-Based and Mean-Shift. As can be seen, the final voted image (V) is a linear combination of each partial result. Thus, it is only needed to find the weights for this sum.

In order to do this, the parameters computed in section 4 were kept in place and then each algorithm was ran independently on a batch of images from the dataset. For each image, the program computed the L value and chose the weights in an inverse proportional manner to the sum of all L values.

To better explain this, a simple example will be presented: considering that there are 3 algorithms A, B and C with their L values $L_A = 1$, $L_B = 1$, $L_C = 3$. Since a small L values means a better result, w_A and w_B need to have a greater weight than w_C.

The formula used is the following: $w_i = \frac{\frac{1}{L_i}}{S}$, where $S = \sum_i \frac{1}{L_i}$.

For our example, this leads to $w_A = w_B = \frac{\frac{1}{1}}{\frac{1}{1}+\frac{1}{1}+\frac{1}{3}} = \frac{3}{7}$ and $w_C = \frac{\frac{1}{3}}{\frac{1}{1}+\frac{1}{1}+\frac{1}{3}} = \frac{1}{7}$.

Using this formula, on a batch of 150 random images from the dataset, results into the following weights: $w_R = 0.258$, $w_S = 0.253$, $w_G = 0.273$ and $w_M = 0.214$. These are the values we use when applying the linear sum formula to compute the final segmentation image. In figure 8, the weights computation and how they change at each iteration can be seen, starting from all of them being equal to 0.25 and reaching the values presented above.

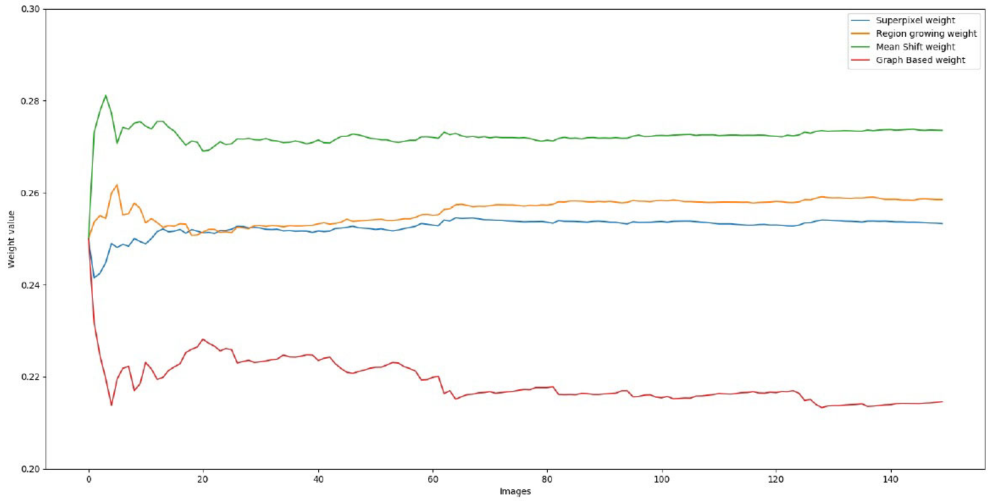


Figure 8. Weights computation

The image computed from the results of all the weighted sums might contain pixels that are from the same region, but have a slight variation in intensity. So, another instance of K-means will be run on this image in order to obtain a final segmented image.

6. RESULTS

In this chapter, the results of running the voting techniques on several test images will be presented. The voting schemes performed well in general, giving good results on various images. In every picture it will be showcased the output of each segmentation algorithm, as well as the final output of the two voting algorithms.

In Figure 9, 10, 11 and 12 the results of running the voting algorithms on images from the dataset can be seen.

The algorithms were ran on 150 random pictures from the dataset and the following mean loss values were obtained:

Algorithm	Mean L value
Region-Growing segmentation	42156.7879
Superpixel segmentation	42409.3569
Graph-Based segmentation	52854.0294
Mean-Shift segmentation	39288.1205
Region Based voting	43254.0269
Weights Based voting	41479.8401

So, as seen from the table, the voting algorithms often give good results in comparison to the labels from the dataset.

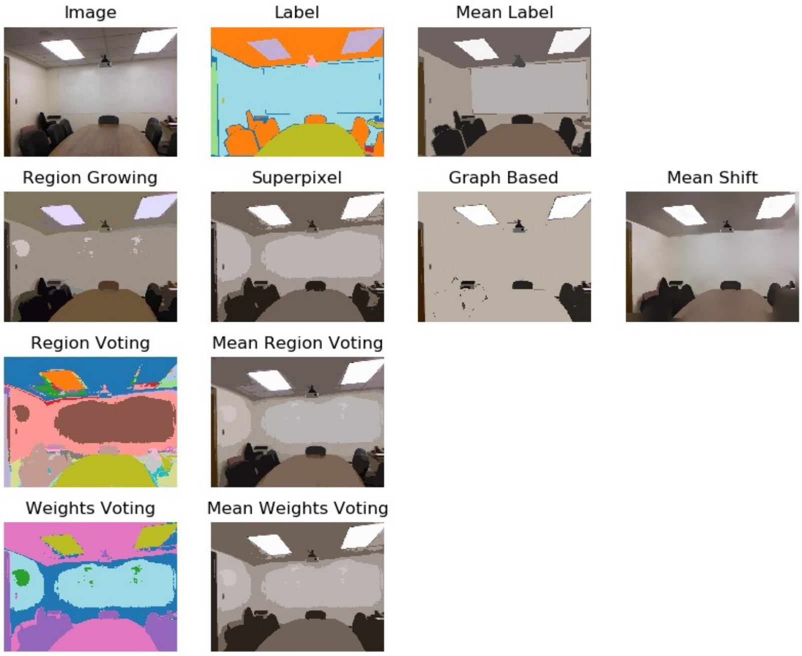


Figure 9. Test execution on image 1

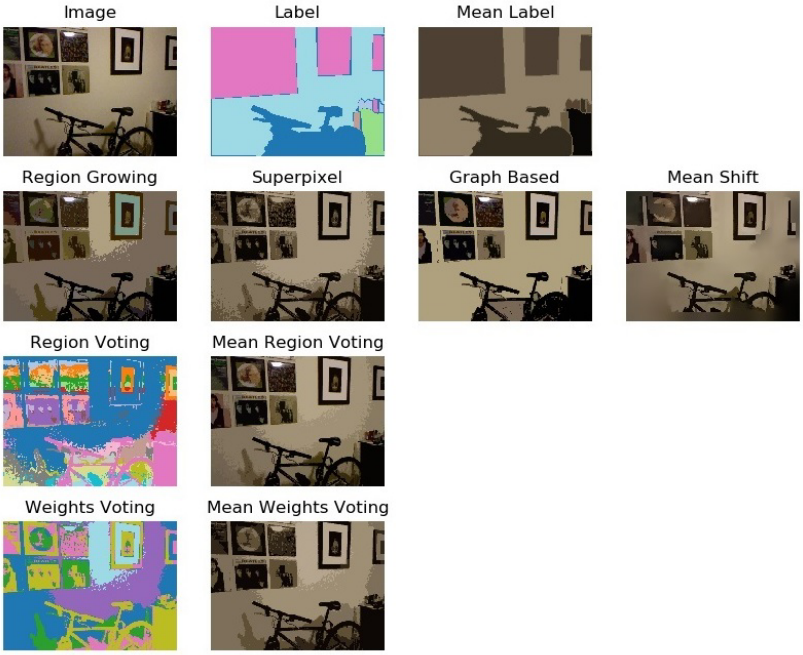


Figure 10. Test execution on image 2

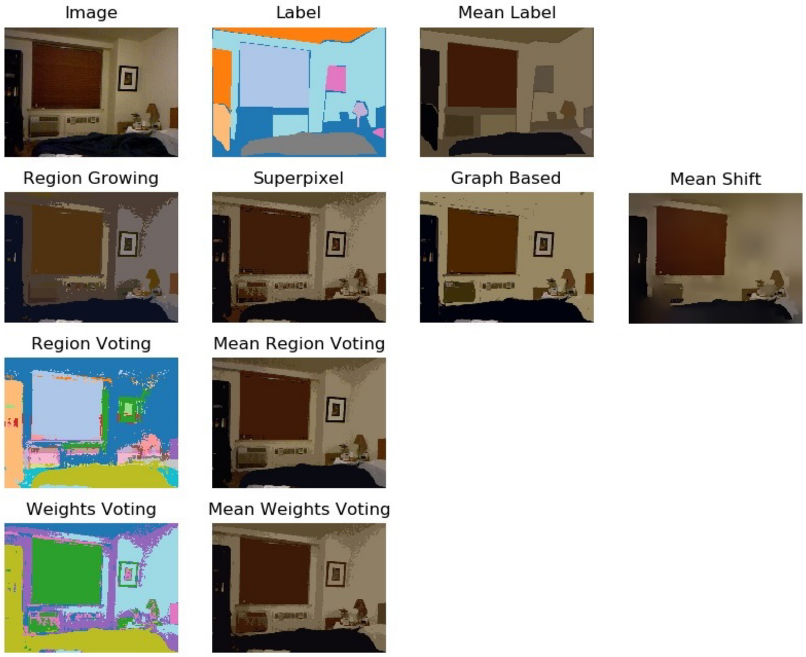


Figure 11. Test execution on image 3

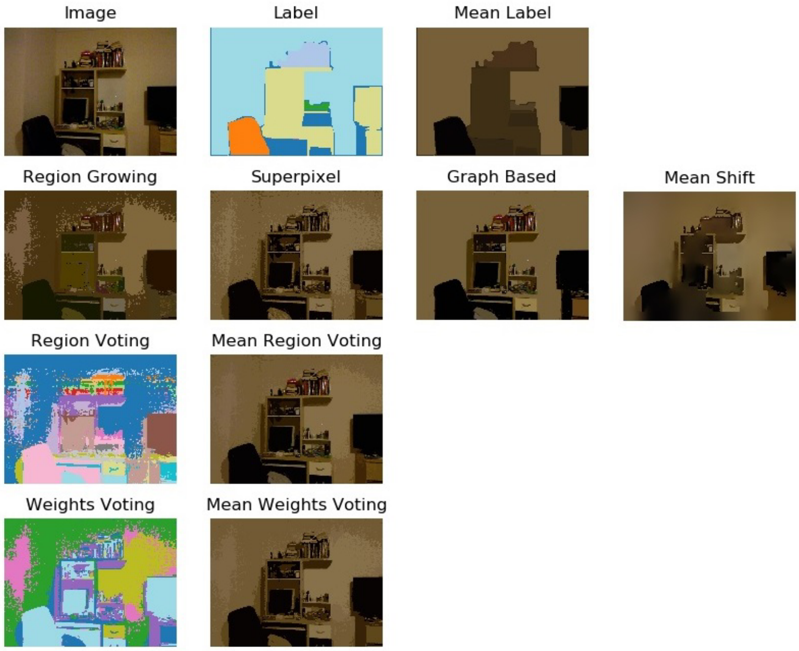


Figure 12. Test execution on image 4

7. CONCLUSIONS AND FUTURE WORK

The voting algorithms presented in this paper are promising and they yielded good results on the tested images. Therefore, it can be concluded that using a voting scheme to combine the results of various segmentation algorithms is viable.

As future work, other segmentation algorithms can be implemented to be used in the voting scheme. Also, a voting method that is based on region indices as presented in [2] can be used. Furthermore, other color spaces can be used during the segmentation process such as the CIE-Lab color space which is more adequate for detection of similar pixels as seen by the human eye. Also, different voting-based methods like those presented in [9][10][11] may be adapted to be used in an image segmentation scenario.

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 si a patrimoniului cultural prin tehnologii avansate" / "Revitalizing Libraries and Cultural Heritage through Advanced Technologies", within PNCDI III.

REFERENCES

[1] L. N. Fred, Finding consistent clusters in data partitions, Lecture Notes in Computer Science vol. 2096, Springer-Verlag, London, pages 309-318, 2001

[2] Costin-Anton Boiangiu, Radu Ioanitescu, "Voting-Based Image Segmentation", The Proceedings of Journal ISOM Vol. 7 No. 2 / December 2013 (Journal of Information Systems, Operations Management), pp. 211-220, ISSN 1843-4711

[3] Y. L. Chang, X. Li: "Adaptive Image Region-Growing, IEEE Transactions on Image Processing", vol. 3, no. 6, pp. 868-872, 1994

[4] Achanta, Radhakrishna, et al. Slic superpixels. EPFL-REPORT-149300. 2010

[5] Efficient Graph-Based Image Segmentation - Pedro F. Felzenszwalb, Daniel P. Huttenlocher (2004)

[6] D. Comaniciu, P. Meer, "Mean-Shift: a robust approach towards feature space analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, IEEE Computer Society, Washington DC, pp. 603-619, 2002

[7] Kanglai Qian, Mean-Shift Segmentation using OPENCV, Available at: https://github.com/qiankanglai/opencv.meanshift - Accessed on: 22 January, 2018

[8] Nathan Silberman, Datasets, Available at: https://cs.nyu.edu/~silberman/datasets/ - Accessed on: 22 January, 2018

[9] Costin-Anton Boiangiu, Radu Ioanitescu, Razvan-Costin Dragomir, "Voting-Based OCR System", The Proceedings of Journal ISOM, Vol. 10 No. 2 / December 2016 (Journal of Information Systems, Operations Management), pp 470-486, ISSN 1843-4711

[10] Costin-Anton Boiangiu, Mihai Simion, Vlad Lionte, Zaharescu Mihai – “Voting Based Image Binarization” - , The Proceedings of Journal ISOM Vol. 8 No. 2 / December 2014 (Journal of Information Systems, Operations Management), pp. 343-351, ISSN 1843-4711

[11] Costin-Anton Boiangiu, Paul Boglis, Georgiana Simion, Radu Ioanitescu, "Voting-Based Layout Analysis", The Proceedings of Journal ISOM Vol. 8 No. 1 / June 2014 (Journal of Information Systems, Operations Management), pp. 39-47, ISSN 1843-4711.