Segmentation and Merger of images

Computer Vision

Instituto Superior Tecnico

Authors
João Oliveira n^o81642
Mario Torres n^o81001

Professors
José Raul Azinheira
Alexandra Moutinho

November 23, 2018

Contents

1 Abstract

This work focuses on the treatment necessary to autonomously extract information from an areal image. In order to achieve this goal a number of computer vision techniques were used. The most important one was the SLIC method which produces superpixels. In order to complement this method an algorithm was designed to augment its performance. Other classical techniques of computer vision were also used on order to test their performance with said areal images.

2 Introduction

This first part of the article focuses on the use of superpixels (in the following part of the work superpixel will be refered as SP) to process areal images. In order to test and explain what the algorithm does, an image with a less complex structure is shown and tested through this first part. The example image is in gray scale, but when the appropriate time arrives the extension to colored images will be explained.

First lets examine figure 1. Using the SLID algorithm complemented with something we would like it to select the coins in isolated SP and the rest of the background to be merged in one unique SP.

Figure 1: Coins on a clear background

Using MATLAB we can use the SLID algorithm directly. If you would like to further understand the meaning and how the algorithm works check [1]. This function allows for two types of methods, the *slic0* and the *slic*. The first one selects automatically certain parameters, so we have less control on the end result while the second one allows for the definition of certain parameters like *Compactness* and *NumIterations*. We will focus on the second method since it gives us more control of the end result. The number of desired resulting SP can also be defined in both of the previous methods.

The first important effect that should be studied is the effect of the *Compactness* and the number of desired SP. By fixing one parameter and varying the other we can see the effect of the *compactness* in 2 and the effect of the number of desired SP in 3.

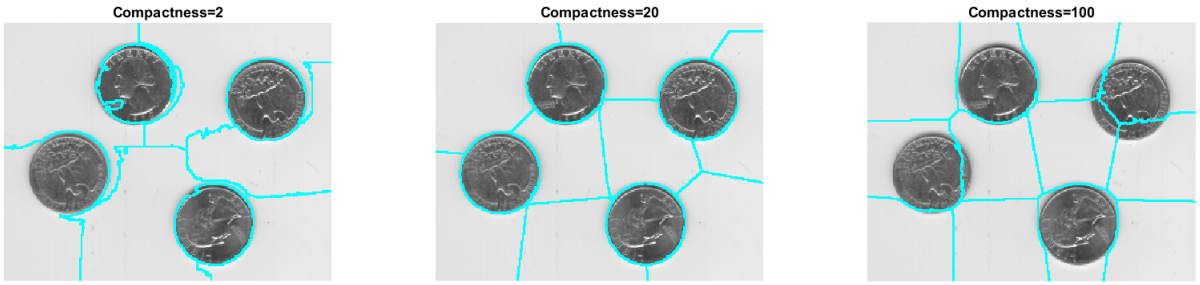

Figure 2: Result of varying the *Compactness*.

The parameter *compactness* controls the shape of SP. A higher value makes the SP more regularly shaped, that is, a square. A lower value makes the SP adhere to boundaries better, making them irregularly shaped.

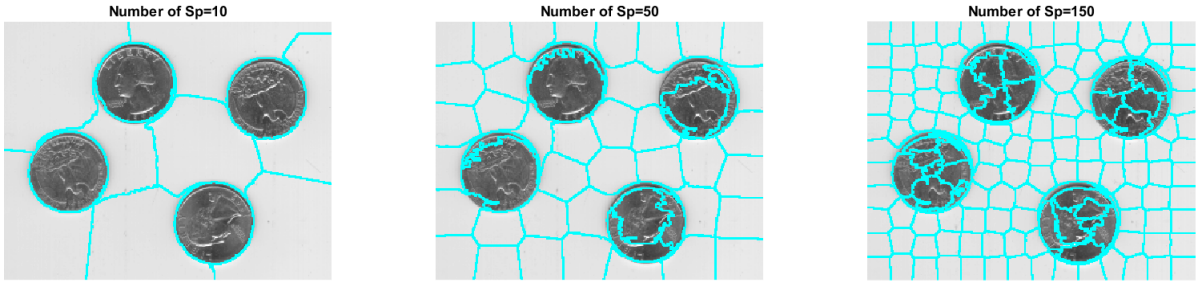

Figure 3: Result of varying the number of desired SP.

The result of the number of desired SP is obvious, although we should mention that we don't necessarily obtain the desired number of SP, the algorithm chooses what number, close to the one inserted that results in the minimum error. In the following sections we will use the values that resulted in the left image of 3, *Compactness*=10 and number of desired SP equal to 10.

3 Adjacent Superpixels

In order to merge any two SP we need them to be adjacent. To determine the adjacency of the SP we developed an algorithm with some hints by kind MATLAB users. To show how the algorithm works on the example image we must first analyze the output of the SLIC method in MATLAB.

In order to reference some superpixel, lets say superxiel one, we will represent it by SP_1. In the following sections this nomenclature will be used.

We obtain two parameters from this output, the first is the number of actual SP in the resulting image, which we will call N, and a matrix L which is a label matrix, which numbers the SP unequivocally. Lets suppose that we want to check the adjacent SP of the SP_1. Looking at figure 4 we can identify that the adjacent SPs of the SP_1 are the number four and the number two.

1	1	1	2	2	2	3	3	3
1	1	2	2	2	3	3	3	3
4	4	2	2	2	2	3	3	3
4	4	4	4	2	3	3	3	3

Figure 4: Example of matrix L

In order to automate this process we select all the positions that are equal to the number 1. From this selection we obtain figure 5.

1	1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Figure 5: Identifying the first SP.

This matrix can be interpreted as an binary image so we can perform a dilation on the 'identified object'. From this dilation and subsequent interception with the conjugate image in figure 5 we obtain the border pixels of the image. This result in shown in figure 6.

0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Figure 6: Frontier with neighbours SP.

Now we can just select the elements in the resulting matrix from L and we obtain the SP numbers from this matrix.

The result of this algorithm is the graph in figure 7 where each node represents a SP and the edges of the graph represent which SP is adjacent to which. By simple inspection we know that the SP_4 is connected to the SP_2, so we would need to run the previous algorithm for all SP in order to find all the connected neighbors.

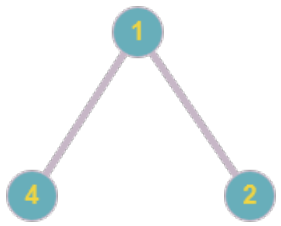

Figure 7: Adjacent superpixels of SP_1.

Now moving to the image shown in 1 we can run the same algorithm and see if we indeed obtain

the desired results. First we show with numbers the SP in the image after being processed by the SLIC method (figure 8).

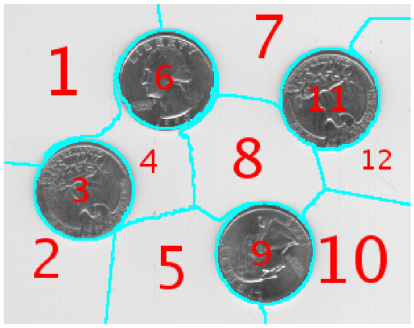

Figure 8: Numbered superpixels.

And the resulting graph in 9. We can see that apparently there is a mistake, the node four is connected to node 7, but in fact the SP_4 involves part of the SP_6, and therefore it has a frontier with the SP_7.

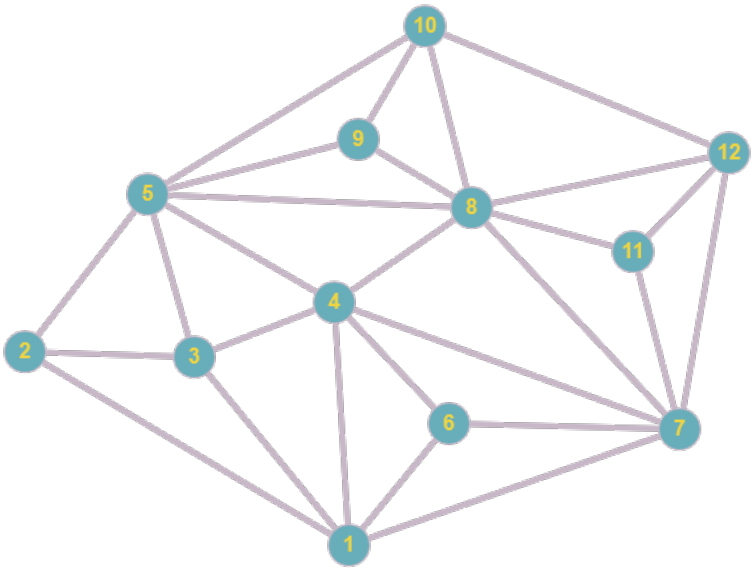

Figure 9: Adjacency Graph.

With this graph we can now perform the merge of the different nodes. In the case of processing colored images the treatment of the image is exactly the same.

4 Node Merge

In order to merge the SP we need to define a criteria that defines those SP that should be merged together and those that should not be merged.

4.1 Criteria for Node merge

In order to obtain the group of SP that should be fused together we define an acceptable region of merger. We decided that a simple threshold should suffice. In the following section the symbol SP_i will represent the superpixel i defined in the label matrix,L, and the letter I_i represents the value of the intensity of the image in the gray scale case and in the case of a colored image it represents a vector with the values of the intensity for each channel of the image.

In the case of a gray scale image, the SP_i should be fused with the SP_j if and only if they respect the condition

$$|I_i - I_j| \leq T$$

where T represents the value of the threshold that we wish to respect.

In the case of a colored image a natural extension would be to merge SP_i and SP_j if their Euclidean modulus in the color space that they are represented, we assume it to be RGB, is smaller than some threshold.

$$(I_i(r) - I_j(r))^2 + (I_i(g) - I_j(g))^2 + (I_i(b) - I_j(b))^2 \leq T$$

which would result in a selection in tree dimensions like figure 10.

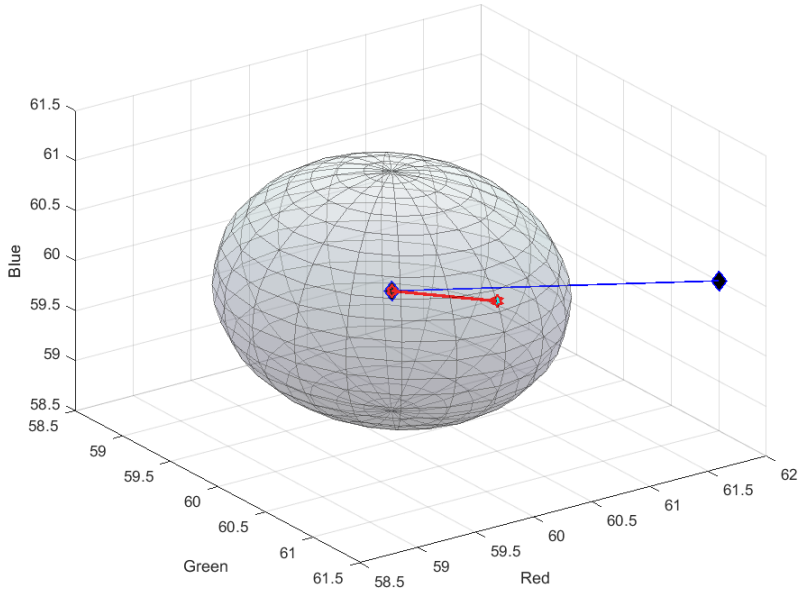

Figure 10: The Euclidean norm of the two red SP is smaller than T but the same is not true for the blue SP so they are not merged.

For different types of color spaces the conditions would probabily need to be defined in a toroidal space, because the HSV is defined by a cone, this means that the value of H equal to 1 is the same in terms of representation as the value 0.

4.2 Group Merge

Due to the way that our solution works there is an advantage compared to other methods.

Lets suppose that SP_i should be merged with SP_j but not with SP_k, but the node SP_j is suposed to merge with both nodes, that is SP_i and SP_k, then all tree of these nodes will be merged. This allows us to merge SPs that were separated due to differences in lighting. This can be explain by looking at figure 11.

Another algorithm dealing with this image might separate all these pixels, but if the difference in lighting is small, then they might all belong to the same object.

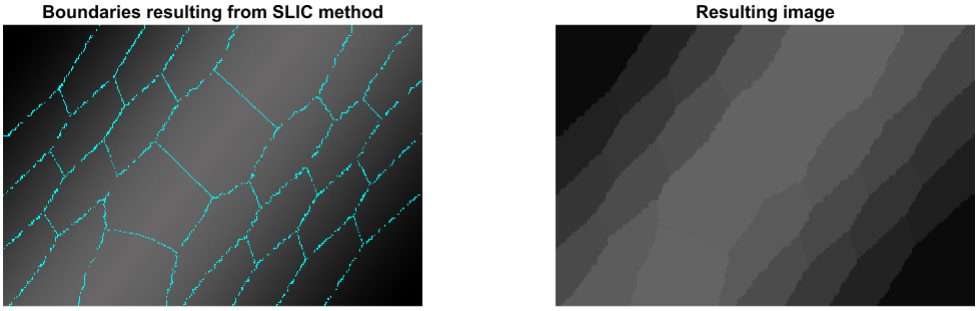

Figure 11: Result of SLIC for lighting differences.

The way we select the group of nodes that should be merged is by iteratively selecting the paths in the graph that should be kept and the ones that should be eliminated. In the end we will obtain separated graphs, and the nodes in these separated graphs should be merged.

Returning to the previous example of figure 1 we can run the algorithm and see that the resulting graph is presented in 12.

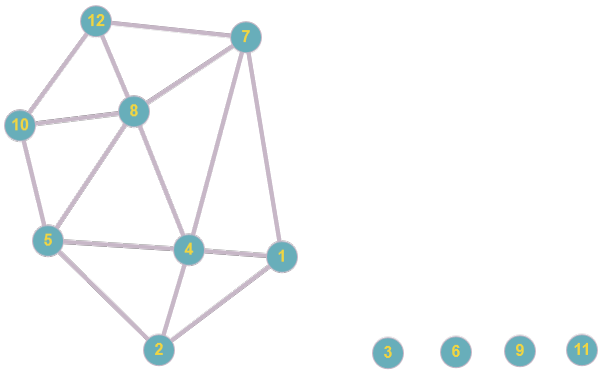

Figure 12: Result of the developed algorithm.

If we compare against figure 8 we see that the desired result was obtained. The visual result can be seen in figure 13.

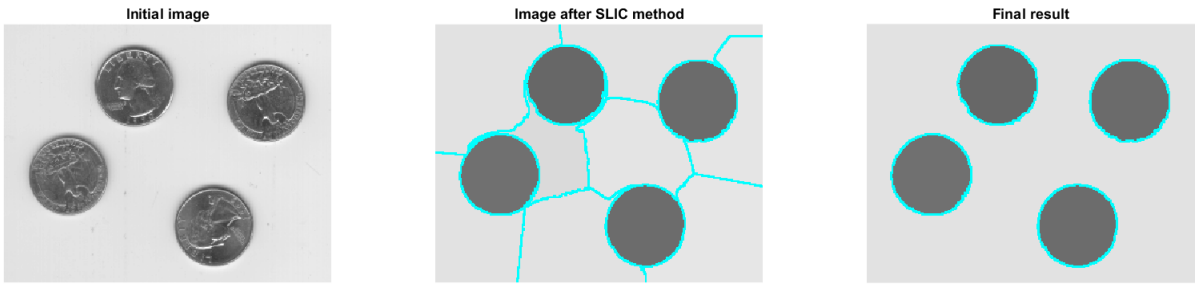

Figure 13: Sub Graphs of different SP that were merged.

5 Activity Diagram of the algorithm

In order the sum up the previous explanation in this section the activity diagram of the algorithm is shown, in case the user of the algorithm does not want to read the previous developed work, although we strongly suggest for you to read the previous sections, since you can have a deeper understanding of how the algorithm was developed.

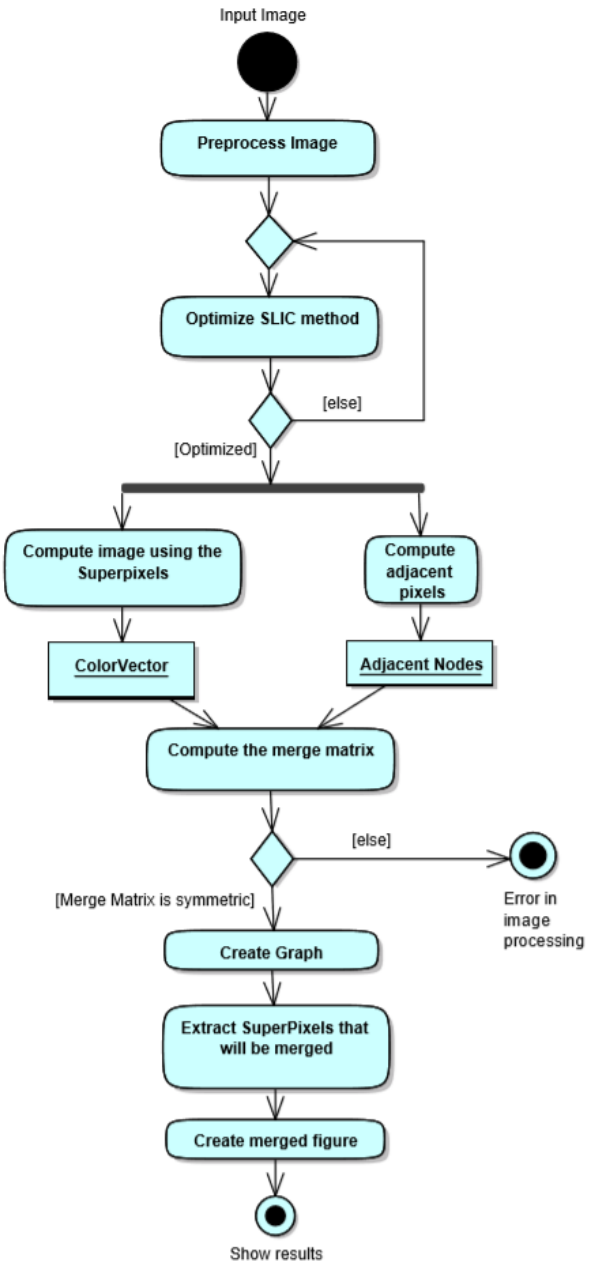

Figure 14: Activity Diagram.

If you have never dealt with activity diagrams check [2]. There is a brief explanation regarding the components of this family of diagrams.

6 Results for the areal images using Euclidean norm

Using the previous developed algorithm we can set the restriction of the maximum acceptable Euclidean norm between two superpixels. In figure 15 we see the image that we are testing the algorithm.

Figure 15: Test image.

Using this image and the threshold set to 10 units in the RGB space the result can be seen in 16.

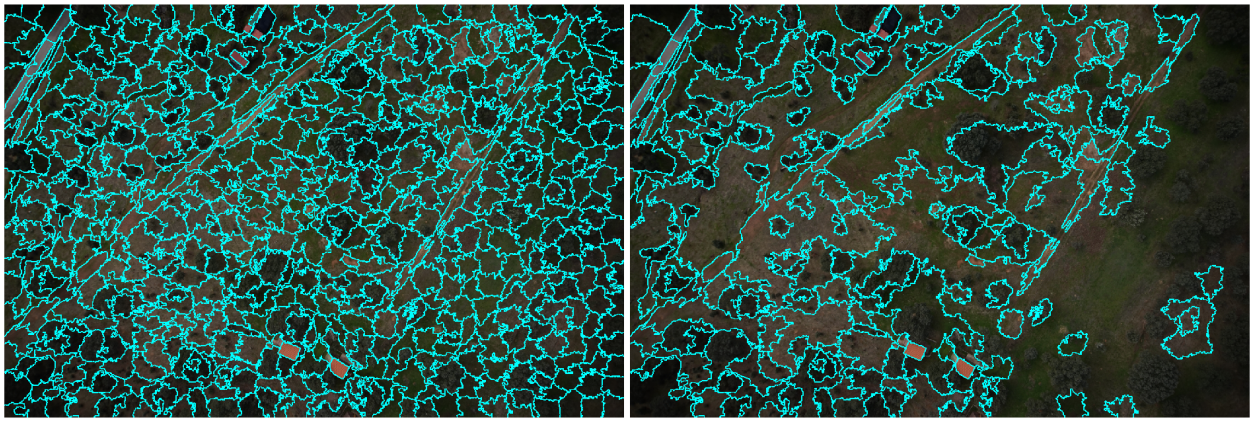

Figure 16: Result of Euclidean threshold with 10 units.

The result is not what we expected. Due to the image being too dark the threshold is not selective enough in some areas, like the trees in the bottom of the image, and the algorithm is too selective in the roads, producing segmented parts. Even if we reduce the threshold these problems will not disappear. One way to compensate would be to increase the contrast of the image. By using power law with $c = 1.5$ and $\gamma = 1.05$ the result can be seen in 17.

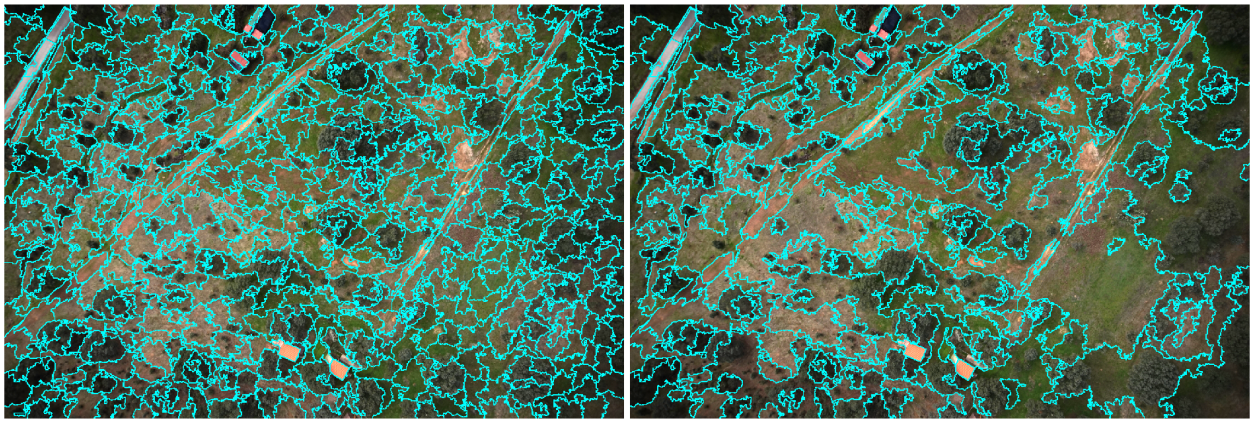

Figure 17: Result of Euclidean threshold with 15 units.

So even with the increase in contrast the result still merges objects that should not be merged and separates SP that should be merged.

These results take us to the conclusion that another method should be used regarding the merge using color information. This new method is presented in the next section.

7 Node merge with invariance to lighting

In RGB the color of two pixels is the same if a strait line passing though the origin contains the two pixels. This is ilustrated in 18.

Figure 18: Different colors for vectors with same direction but increasing norms.

So in order to better select information regarding color only we should use the angle between the two vectors in RGB of the SP in order to determine if we want to merge the mentioned SP or not.

We can now ponder how might we add this new criteria to the previous developed work but the answer is simple, the algorithm works as long as the criteria selected is invariant regarding the order. Assuming that f is the function that determines the merger of two nodes, then the merge graph can be constructed as long as $f(SP_1, SP_2) = f(SP_2, SP_1)$.

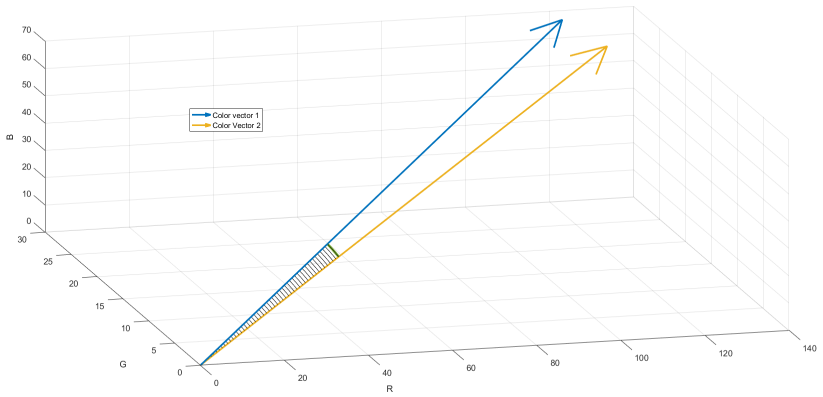

Figure 19: Angle between two vectors in RGB space.

7.1 Results with invariance to light

Considering the image presented in 15 we can apply the merge condition with the angle between the two superpixels in the RGB space.

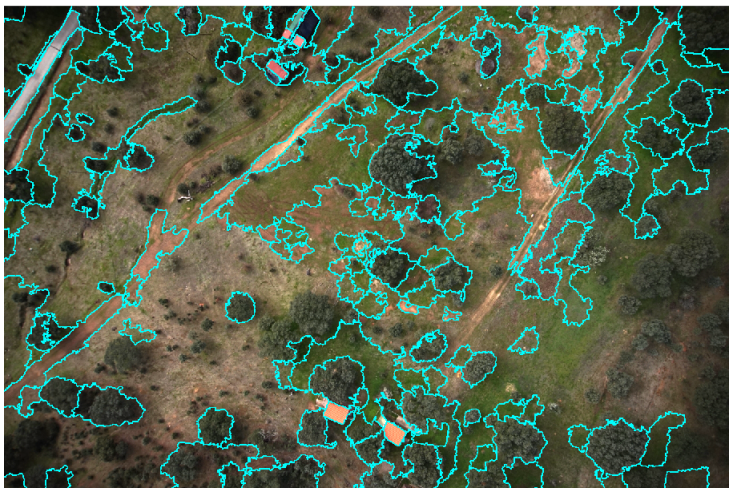

Figure 20: Merge result with angle equal to 0.04 radians.

By visual inspection we can conclude that the algorithm isolates reasonably well the objects in the image, but it is too permissive, some trees in the left side of the image were merged, due to their median color being similar to the background (the soil). If we think of correcting this problem using a smaller angle for the merge condition the final result will contain areas that should have been merged and they were not, as it can be seen in 21.

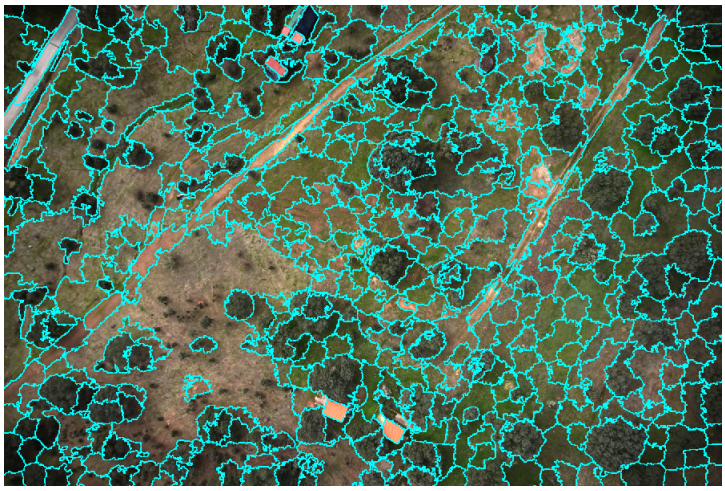

Figure 21: Merge result with angle equal to 0.02 radians.

To correct the permissiveness of the algorithm a complementary condition can be used. In the next chapter the inner workings of the algorithm will be explained as well as the way in which it complements the angle criteria presented in this chapter.

8 Histogram analysis

Since we want to complement the previous condition we developed a statistical analysis of the objects that we wish to remain segmented once the merge occurs.

In order to do this we decided to analyze the histogram of the trees, roads, water and the ground. These are the main components that we know the previous condition is too permissive on. There are three statistical tools that we use, the standard deviation, skewness and the kurtosis.

To obtain this data for the different objects in the image one first needs to perform a selection by hand. For the images provided we obtained the regions shown in 22.

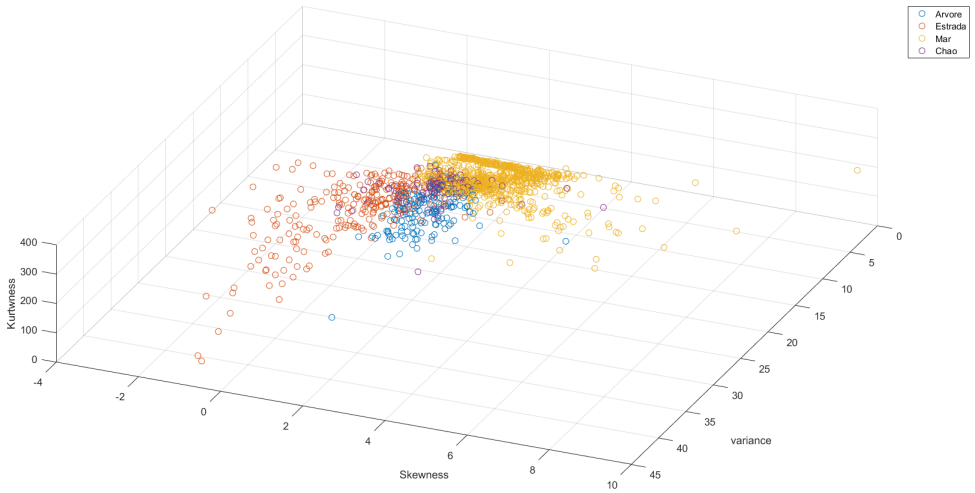

Figure 22: Statistical distribution of the objects in the areal images.

Now with these points classified into different classes we can design a probability of a certain SP to belong to any one of the classes defined using a Gaussian distribution.

Assuming that $x = [x_1, x_2, ..., x_n]$ is a vector valued random variable with a multivariate normal distribution with mean $\mu \in \mathbb{R}^n$ and a covariance matrix $\Sigma \in S$ then its probability density function is given by

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{(-\frac{1}{2}(x-\mu)^T(\Sigma)^{-}1(x-\mu))}$$

Now if we want to merge two SP we just need to compute the class that this SP belongs and if the result is the same then they should be merged.

9 Final results using the histogram statistics and the color condition

By merging the two previous conditions with an AND gate we can compensate for the weaknesses of both conditions. The results in 23 and 24.

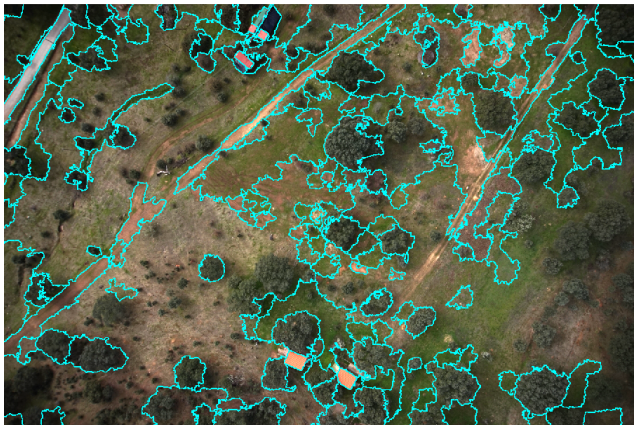

Figure 23: Final result for example image 1.

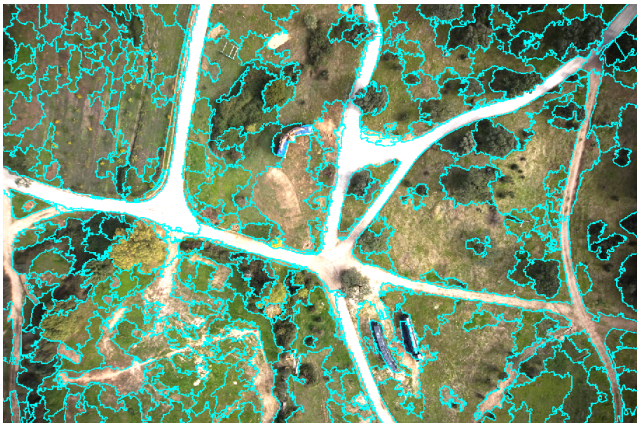

Figure 24: Final result for example image 2.

By performing a optimization of the parameters the results can be further improved and the algorithm can be adapted to different types of image to preprocess.

10 Future Work

To construct a full fledged paper which could be published and bring new ideas to the table we had a couple more corrections to the previous model, such as correcting the errors present in the statistical analysis. One way to do so would be to construct a voting mechanics where we would retrieve the certainty of each individual method and by majority voting with some threshold we would decide if the SP belongs to the same object or not. The difference from the previous DC (Decision Criteria) lies in the the use of the AND gate. If one of the methods determines that two SPs don't belong to the same object then they will not me merged. Obviously we are introducing a rigidity in the decision criteria that could be softened by introduced the majority voting.

Another interesting idea would be to first run the algorithm without performing the merge of the SPs, but instead save on the nodes which class should be attributed to them. After this stage another algorithm that would check for the size parameters of each object in the image, lets say for simplicity the average number of pixels for a tree, and change the class on the neighborhood nodes until the size of that object is respected, and this process would be run on all the objects for all classes.

Another interesting idea that we devised taking inspiration from deep learning was the use of a filter, that resizes to a smaller image, and performs an weighted average in the previous one.

By performing successive iterations until a small uniform image is obtained with different uniform objects and then perform the inverse process of recreating the initial image of the same size. The resulting image would theoretically show all the objects of the initial image segmented as uniform blobs.

11 How to use the GUI

For a more user friendly testing of parameters the previous methods were implemented in a GUI application. The app begins by running the script StartHere.m and from there the first tab window is presented with a list of the necessary files to be present in the program folder GUIuser. Any images that are to be used and analyzed by the app must be in the Images folder in jpg format.

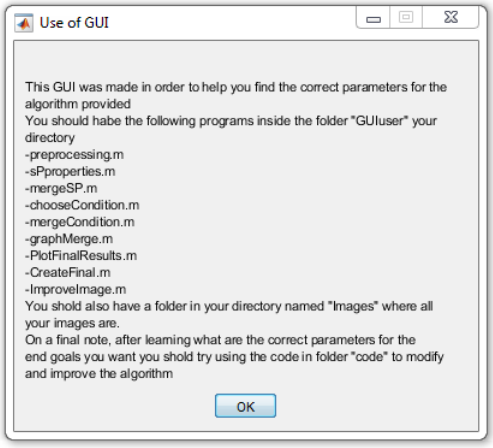

Figure 25: First Window tab

The interaction with the GUI is straightforward and from the second tab the user begins by selecting an image from the Directory list. The preview is automatically shown after selection. When the image is selected the user chooses the parameters for the Superpixel algoritm and the resulting superpixel grid is shown on display.

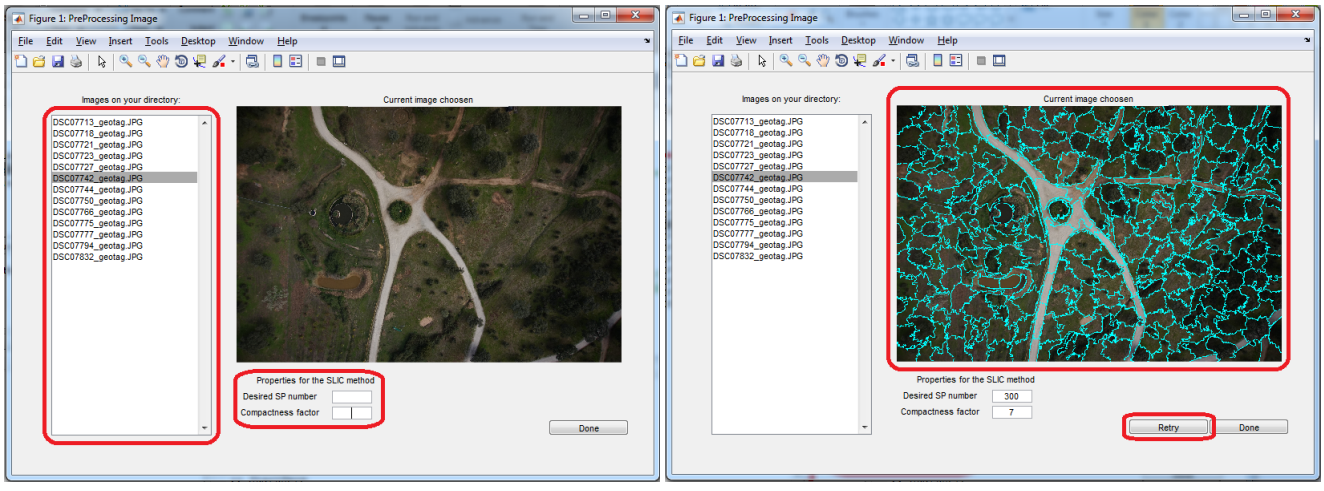

Figure 26: Second Window tab

After this the user can either choose the current Superpixel grid shown or reset the parameters and try again for a better result. The next stage is to choose the method to merge Superpixels. Currently there are four methods available, the same methods shown in the previous chapters.

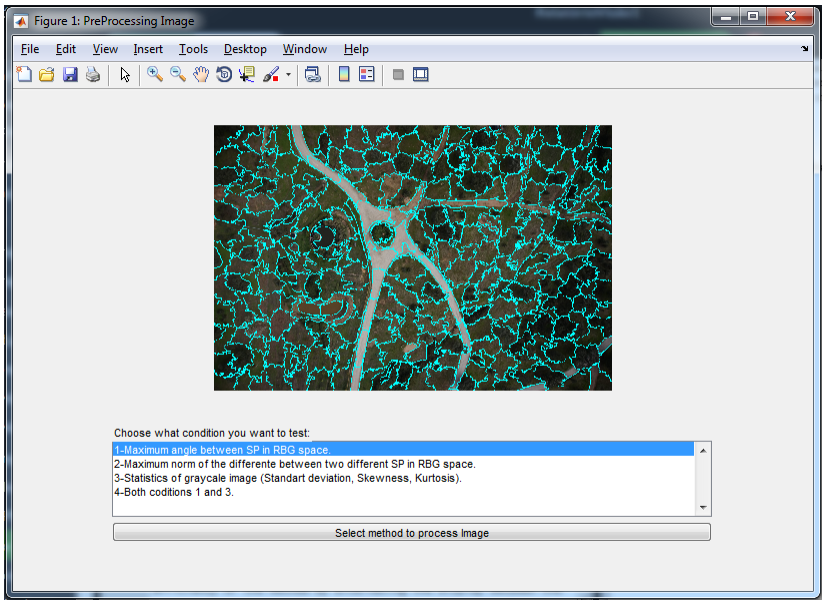

Figure 27: Fourth Window tab

After the merging is computed the user can choose to compare the efficiency of the method by alternating the display between the pre-processed image and the resulting image. There is also the option to view represent each superpixel median color in the image by clicking the button Median Color and then which superpixel grid wants to see, pre-processed or the result image.

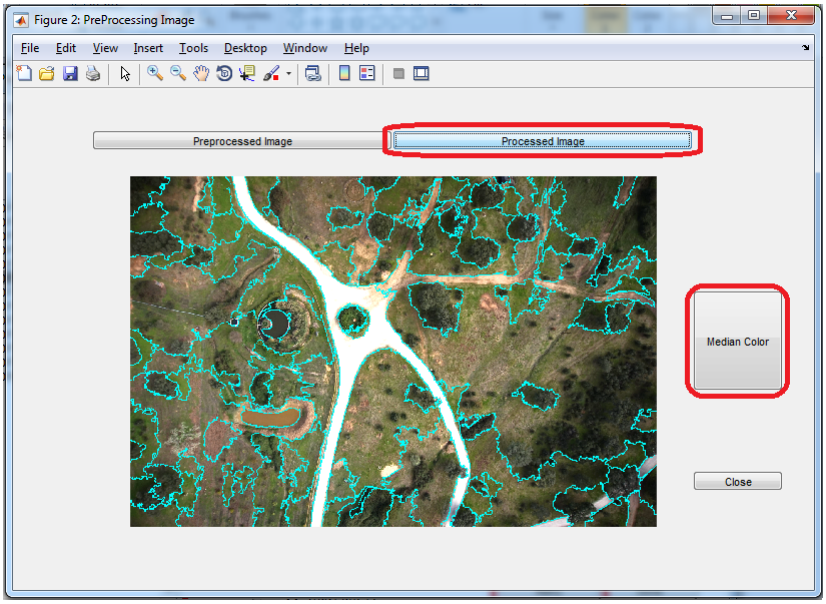

Figure 28: Final Window tab

Bibliography

[1] https://www.mathworks.com/help/images/ref/superpixels.html

[2] https://www.lucidchart.com/pages/uml-activity-diagram