Scene Segmentation and Interpretation

Lab report: Image segementation algorithms

Eduardo Ochoa & Roger Pi

March 13, 2018

1 Introduction

Image segmentation is still a relevant research area in the field of computer vision. It is a domain that consist on partitioning an image into a set of non.overlapped regions that are visually different, homogeneous and meaningful with respect to some characteristics. As important as it is and the many applications that has in different fields such as in medical image analysis, computer vision, image pre-processing or other domains; it is important for us as students on this field to learn not just the main aspects and methods that exist but to implement them.
In this lab-work a Region Growing algorithm has been implemented using Matlab framework. Using a set of test images the implementation of the algorithm has been tested with gray and color images. Furthermore, we contrasted the algorithm with another segmentation algorithm called **mean shifts** which will let us compare the accuracy of each algorithm, their advantages and drawbacks.

2 Algorithm analysis

2.1 Region growing

Region growing is an algorithm for image segmentation in which neighboring pixels are examined and added to a region class if no edges are detected. This algorithm belongs to the set of region based methods in which we segment the image into homogeneous areas that are connected through the pixels and meet a certain criteria or characteristic. The implementation of the algorithm should follow the following steps:

- First, we put an arbitrary seed into our image and start comparing with the neighboring pixels.
- The region then starts to grow depending on which the neighbor pixel meets the criteria that have been chosen as an evaluation.
- If a neighbor pixel is added into the region then it's neighbors has to be evaluated to see if they do not belong to the area too.
- The region stops growing when there are no more connected pixels to the current are that meet the criteria. Thus if all the pixels has not been classified we put another seed into our image and start growing a new region.
- The algorithm continues until all pixels belongs to a region.

Looking at this algorithm it can be observed that the implementation should consider different kind of parameters that may result in different kind of solutions. First of all, the number of seeds that we have to place in our image and where to place them should be decided. This, brings another problem which is, having a seed placed in the image how are we going to explore our pixels, what kind of connectivity we will use and what kind of effects will have this in our regions. Finally, we have to consider the criteria that it will be used to decide if a neighbor pixel should be added or not into our region and how we can update this criteria to make it more representative of our areas.
This decision have been taken into our function design and we gave some kind of freedom to the user to set some of these parameters and for evaluate what kind of results fit best to his application.

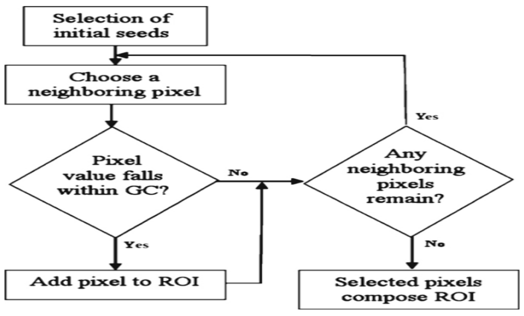

Selection of initial seeds
Choose a neighboring pixel
Pixel value falls within GC?
No
Yes
Add pixel to ROI
Any neighboring pixels remain?
Yes
No
Selected pixels compose ROI

Figure 1: Region growing algorithm flowchart [1]

2.2 Mean shift

Mean shift is another clustering technique very known and used because it has several nice advantages. It tries to estimate the underlying distribution for a set of data.

Mean shift works by placing a gaussian kernel on each point of the data set. Adding all the kernels it creates a probability surface.

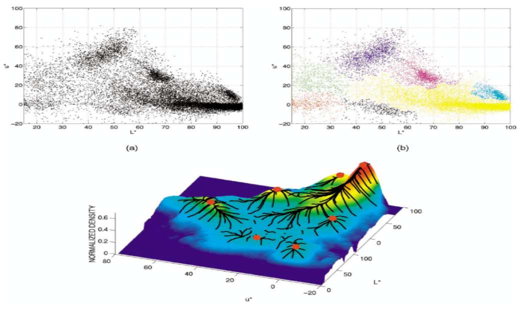


Figure 2: Probability surface

Mean shift works with the probability function as if the points were trying to climb to the nearest peak of the surface, so depending on the kernel bandwidth parameter used, the resultant density function will vary. It will iteratively moving (shifting) each point until it reaches a pick. The algorithm will finish when the average displacement of the points is less than some threshold.

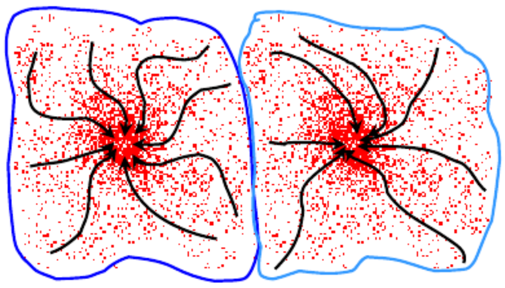


Figure 3: trajectory representation to the center of mass

3 Design and implementation

3.1 The region growing implementation: Explained

As explained in previous section our algorithm implementation needs to be initialized with some important parameters: seed placement, criteria and threshold and pixel connectivity. This was done in Matlab software and tested over in gray scale images and RGB ones. Our function is called by **region_growing(image, threshold, connectivity)** and it expects three arguments to be run. In fact, the proposed implementation let us use the function just by giving an image as an argument and it initialize the other two with some defaults.

Inside our function we compute the regions of the image and return two variables **new_image and number_regions** which give us the segmented image and the total of regions that were found by the algorithm. The seed placement is been fixed in our process by looking at the new image array and finding the first empty pixel (pixel with zero value that means that has not been assigned yet), so for the first region this will be the pixel with coordinates $(1, 1)$.
Then, starting from this pixel a queue is created to explore the neighbors pixel that belongs to the region until there are no more elements in the queue, which means that there are no more connected pixels that meet the criteria or homogeneity of the region. The criteria used is the mean intensity value and in each iteration the neighbor intensity is compared with this value and a threshold, if the difference of intensity is less or equal than the threshold then is added to the region and to the queue for further exploration. Whenever a new pixel is added the criteria is updated to represent the area in totality. Finally, when one iteration is finished then the algorithm looks for the first pixel in our new image array that has zero value and restart the queue. This process is been done until there are no more free pixels in our image.

As it was said the threshold and the connectivity (4-pixel or 8-pixel connectivity) parameters could be given by the user to test different kind of result. If this parameters are not given they are fixed by our algorithm in default parameters, 50 value for threshold and 4-pixel connectivity. Furthermore, the user have to input a double type image that can be either RGB or grayscale value and get the results.

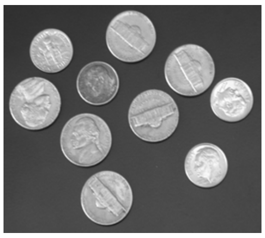
3.2 Mean shift implementation

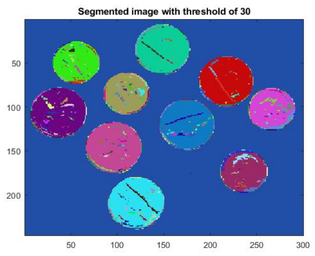
The mean shift algorithm [2] we will use to compare results was found in the website Matlab File Exchange and implemented by Yue Wu. Complete link can be found on the matlab test file.

4 Results

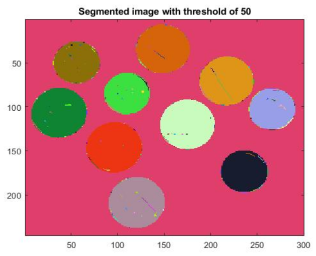
4.1 Region growing results

The region growing implementation was tested with a gray-scale image and was improve in terms of speed during our tests (improvements that are detailed in the following sections). The basic trials that were done consist in changing the threshold parameter and test this thresholds with both connectivities. Then, we compared both connectivities in term of speed and number of regions that were classified. In image 4 we are able to see the original image that was used in our algorithm, along with the 8-pixel connectivity set as a parameter to the function we start changing the threshold parameter in values of 30, 50, 100 and 150. By looking at the results we can notice that a low threshold value, as in 4b, generate an over-segmented image with a lot of regions but this is preferable than having an image as 4e in which almost all the coins have disappeared due to the high threshold value in which we are saying to the algorithm that let a lot of pixel in one region even when in reality they do not share the same characteristic. By testing and looking at 4c, it is preferable to have a threshold value that is in-between of having an over-segmented image and a under-segmented one.

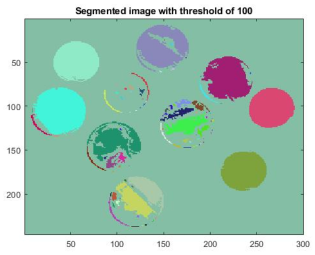

(a) Original coins image



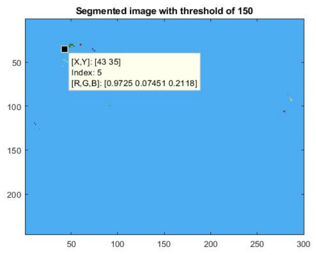
(b) Threshold of 30



(c) Threshold of 50



(d) Threshold of 100



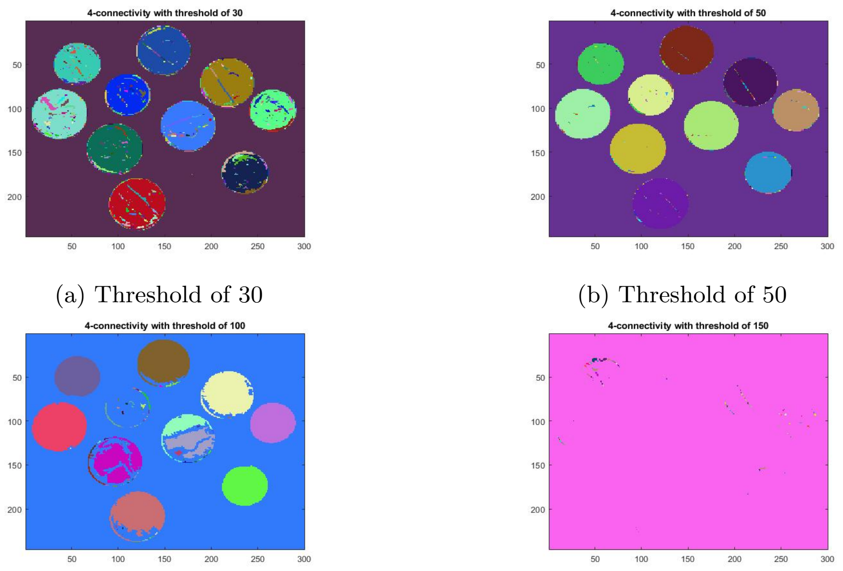
(e) Threshold of 150

Figure 4: Region growing algorithm implementation with 8-pixel connectivity and different threshold values

Then, we compared this results with an 4-pixel connectivity parameter given to the function by applying the same threshold values to the image. The results are shown in figure 5. By visual inspection is hard to notice any difference between both connectivities, but this comparison can be seen in Table 1 in which a contrast in speed and number of regions generated is shown. It is possible to observe how the 4-pixel connectivity generates a lot more of regions for each threshold value but in terms of speed performs better. Due to the speed of the 4 connectivity we can say that is better than the 8 connectivity one, it is true that more regions are detected but most of the time over-segmentation is preferred than having a few regions and in terms of cost of computation we save a little bit more with this connectivity. Also, as it is depicted the threshold value of 50 was the one that performed the best.

Threshold value \ connectivity	8-connectivity	4-connectivity
30	846 regions 0.805 secs	1339 regions 0.775 secs
50	318 regions 0.595 secs	494 regions 0.548 secs
100	106 regions 0.559 secs	114 regions 0.483 secs
150	27 regions 0.485 secs	94 regions 0.476 secs

Table 1: Comparison table (number of regions and speed) of pixel connectivity implementation.



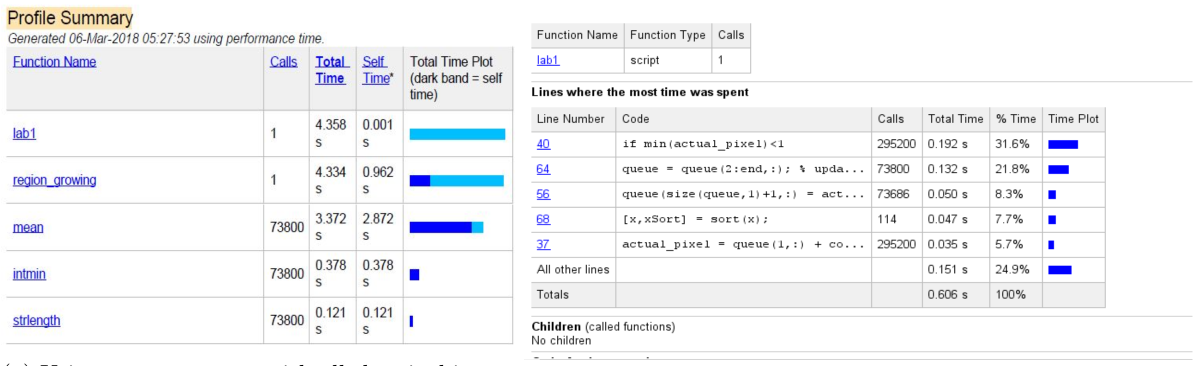
(a) Threshold of 30 (b) Threshold of 50

(c) Threshold of 100 (d) Threshold of 150

Figure 5: Region growing algorithm implementation with 4-pixel connectivity and different threshold values

4.1.1 Algorithm improvements

One major improvement was made on our algorithm. This improvement came in terms of speed of our implementation. At the beginning, and as it is shown in Figure 6, our algorithm was taking nearly *8 seconds* to process an image with a threshold value of 50 and 8-connectivity. This was due to the fact that inside the process a mean vector with all the intensities of the pixels in the region was been kept and then the Matlab function mean was called. The problem was fixed by keeping a counter of the number of pixels that had been integrated to the region and performing a mathematical operation to update the mean.



Profile Summary

Generated 06-Mar-2018 05:27:53 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
lab1	1	4.358 s	0.001 s	
region_growing	1	4.334 s	0.962 s	
mean	73800	3.372 s	2.872 s	
intmin	73800	0.378 s	0.378 s	
strlength	73800	0.121 s	0.121 s	

(a) Using a mean vector with all the pixel intensities

Function Name	Function Type	Calls
lab1	script	1

Lines where the most time was spent

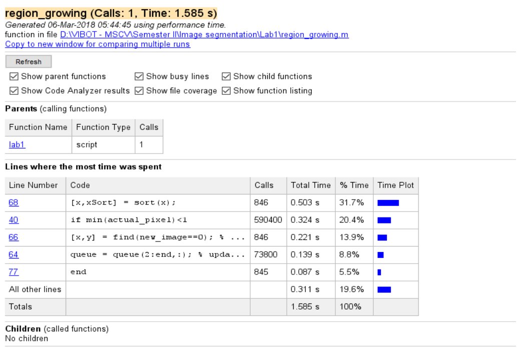
Line Number	Code	Calls	Total Time	% Time	Time Plot
40	`if min(actual_pixel)<1`	295200	0.192 s	31.6%	
64	`queue = queue(2:end,:); % upda...`	73800	0.132 s	21.8%	
56	`queue(size(queue,1)+1,:) = act...`	73686	0.050 s	8.3%	
68	`[x,xSort] = sort(x);`	114	0.047 s	7.7%	
37	`actual_pixel = queue(1,:) + co...`	295200	0.035 s	5.7%	
All other lines			0.151 s	24.9%	
Totals			0.606 s	100%	

Children (called functions)
No children

(b) Mean deduced by mathematical operation

Figure 6: Algorithm speed improvement by replacing the usage of the mean function of matlab by a mathematical operation to update the mean value. The speed was improved from 8 seconds to 0.606 seconds

Another improvement in speed was done by replacing the way in which we found empty pixels in our new image array. At the beginning we look up for empty pixels and sorted the coordinates given by the function **find( )** in Matlab to get the pixel with less values in the row coordinate. This sorting was computational expensive and by testing the 8-connectivity with a threshold of 100 we could see how it was taking *1.585 seconds*. Instead of this we performed a transpose of the array and this did the same result as teh sorting. The improvement on speed was not that big but it went down to *1.354 seconds* as it is shown in figure 7.



region_growing (Calls: 1, Time: 1.585 s)
Generated 06-Mar-2018 05:44:45 using performance time.
function in file D:\VIBOT - MSCV\Semester II\Image segmentation\Lab1\region_growing.m
Copy to new window for comparing multiple runs

Refresh
☑ Show parent functions ☑ Show busy lines ☑ Show child functions
☑ Show Code Analyzer results ☑ Show file coverage ☑ Show function listing

Parents (calling functions)

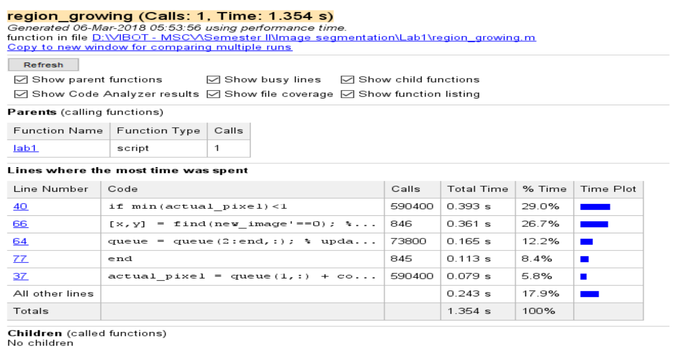
Function Name	Function Type	Calls
lab1	script	1

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
68	`[x,xSort] = sort(x);`	846	0.503 s	31.7%	▬
40	`if min(actual_pixel)<1`	590400	0.324 s	20.4%	▬
66	`[x,y] = find(new_image==0); % ...`	846	0.221 s	13.9%	▬
64	`queue = queue(2:end,:); % upda...`	73800	0.139 s	8.8%	▬
77	`end`	845	0.087 s	5.5%	▬
All other lines			0.311 s	19.6%	▬
Totals			1.585 s	100%	

Children (called functions)
No children

(a) Sort function usage



region_growing (Calls: 1, Time: 1.354 s)
Generated 06-Mar-2018 05:53:56 using performance time.
function in file D:\VIBOT - MSCV\Semester II\Image segmentation\Lab1\region_growing.m
Copy to new window for comparing multiple runs

Refresh
☑ Show parent functions ☑ Show busy lines ☑ Show child functions
☑ Show Code Analyzer results ☑ Show file coverage ☑ Show function listing

Parents (calling functions)

Function Name	Function Type	Calls
lab1	script	1

Lines where the most time was spent

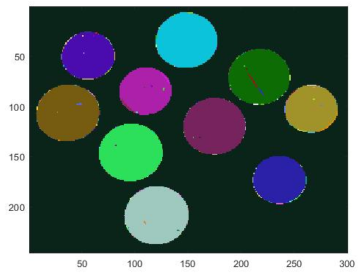
Line Number	Code	Calls	Total Time	% Time	Time Plot
40	`if min(actual_pixel)<1`	590400	0.393 s	29.0%	▬
66	`[x,y] = find(new_image'==0); %...`	846	0.361 s	26.7%	▬
64	`queue = queue(2:end,:); % upda...`	73800	0.165 s	12.2%	▬
77	`end`	845	0.113 s	8.4%	▬
37	`actual_pixel = queue(1,:) + co...`	590400	0.079 s	5.8%	▬
All other lines			0.243 s	17.9%	▬
Totals			1.354 s	100%	

Children (called functions)
No children

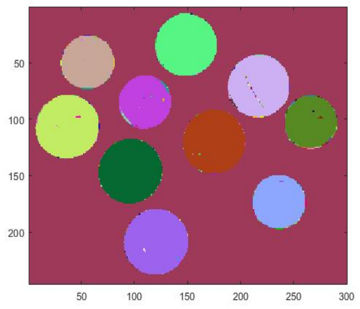
(b) Transpose applied to the image for finding empty pixels

Figure 7: Algorithm speed improvement by applying a transpose to the new image array and find the next pixel in where to place the new seed

At last, one improvement done not to the algorithm itself but to the image was applying a gaussian filter of 0.5 sigma using Matlab function **imgaussfilt()** to take away some noise from the regions and have less regions. This improvement was just to prove that a filter can homogenize a little bit more the pixels and take away some noise. The number of regions from 8-connectivity decreased from 318 to 292 with an applied threshold of 50. And, for the 4-connectivity it decreased from 494 to 406 with the same threshold value. The results are shown in image 8.



(a) 8-pixel connectivity

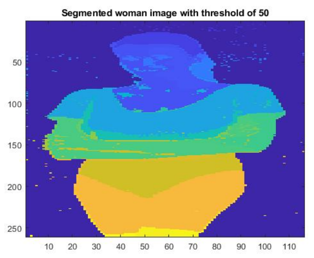


(b) 4-pixels connectivity

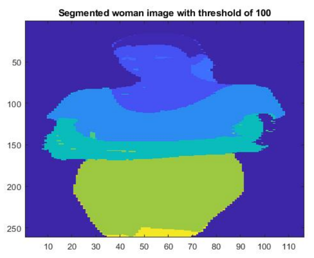
Figure 8: Result of the segmentation process with an input image in which a Gaussian filter was applied

4.1.2 Testing RGB images

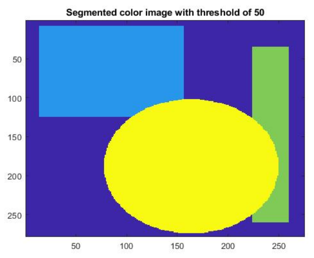
After all the testing and improvements with the grayscale image were done it was just a matter of generalize more our code to be able to read RGB images. This was done just by comparing the size of the image in the third coordinate and if the size was equal to three then a flag was assign with the value of one. This flag is then used inside the algorithm to use instead of the regular mean value a vector of mean values in R, G and B. The results of the implementation tested in three different images are show in Figure 9. The input parameters were a threshold of 50 and 100 with 4-connectivity.



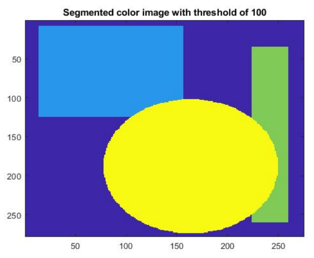
(a) Threshold of 50



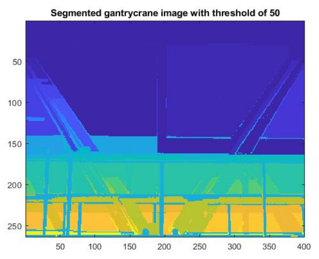
(b) Threshold of 100



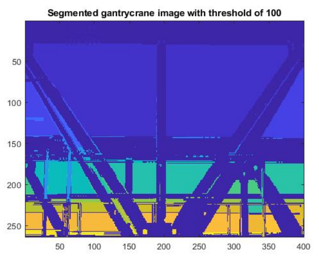
(c) Threshold of 50



(d) Threshold of 100



(e) Threshold of 50



(f) Threshold of 100

Figure 9: Region growing algorithm tested in RGB images

4.2 Mean Shift

Mean shift has been tested over gray-scale and RGB images. One of the advantages of mean shift is that is designed to be used without tuning parameters so much. It needs to specify the kernel used to create the surface and a threshold to decide when to stop. Using a big kernel will make the points move faster because there will be less picks but the surface will be more distributed. But if the threshold is not too strict, some points will not reach the peak, but will get closer, concentrating points somewhere nearer the center of mass.

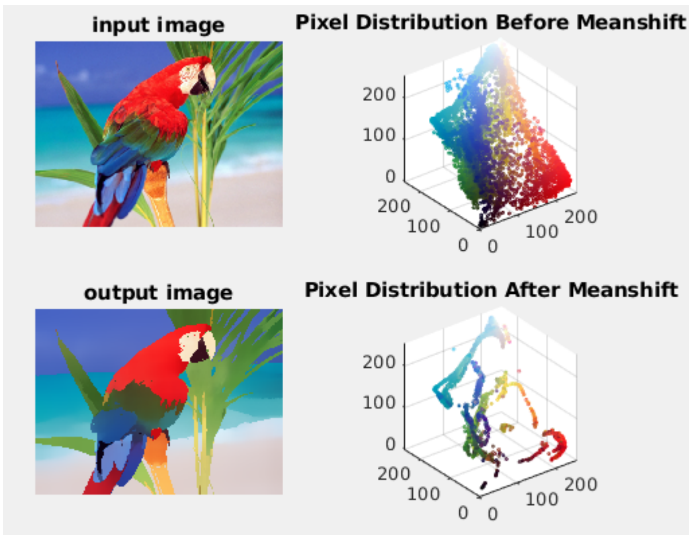


Figure 10: Mean shift clustering applied to test.png

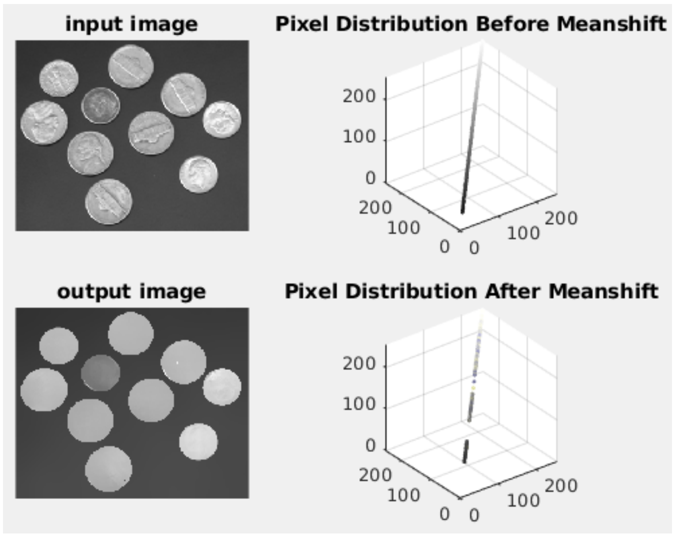


Figure 11: Mean shift clustering applied to coins.png

In Figure 11 the points are represented in 3D using the only dimension repeated in X, Y and Z, so it forms a line.

In mean shift stretch the distribution of the points creating clusters more dense but less distributed. This means that the output pixel value, or region value, preserve some color information, because all them are close to the same pick and their value is around this pick. In other words, as we can see in figure 10 the distribution of the colors is not just some points representing regions, actually its clusters representing regions. For this reason we are representing segmented images with its own color.

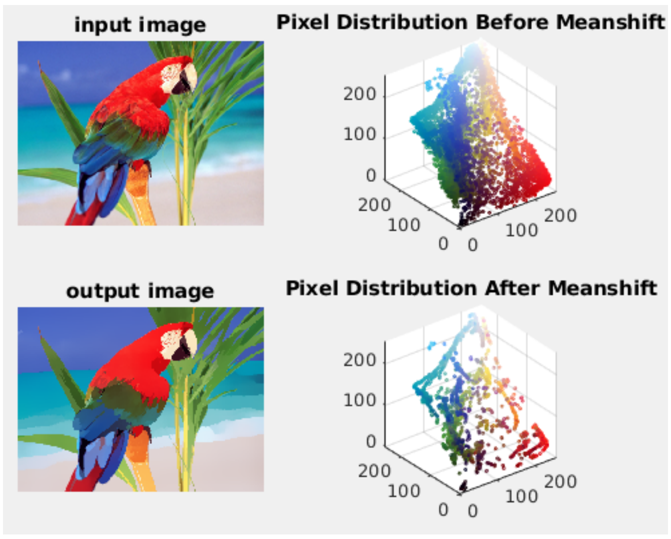
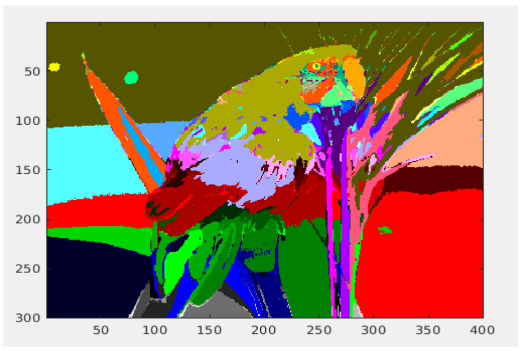


Figure 12: Mean shift with much strongest threshold

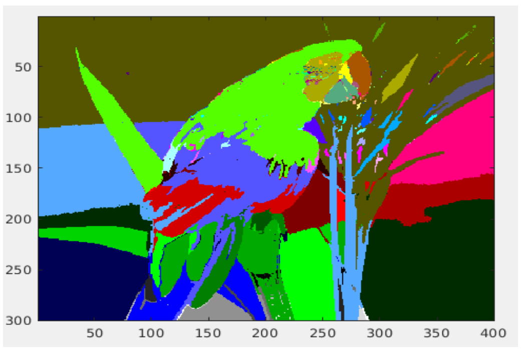
5 Comparison of algorithms

- Speed: Region growing algorithm is much faster than mean shift. While all the tests using region growing were performed under 1 second, mean shift takes between 5 and 20 seconds, depending on how strict do we ask the threshold to be.
- Region tag: Region growing destroy completely all the color information, representing the pixel color as its region tag. Close region values doesn't mean to be similar regions (To do so it should be use its region mean that should be saved as a vector during function run). Mean shift stretch the pixel distribution, saving some color information because it will move close around a probability, but the region is not completely defined by one single tag.

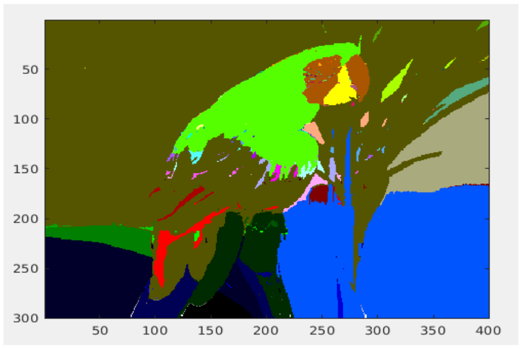
- Number of regions: None of the algorithms (at least our implementation) can predict the number of regions it will be.
- Robustness: Region growing is very sensitive on the threshold value. Also the position and number of seeds will affect to the result. Mean shift creates a more robust result that needs less tuning and will not be different on different iterations with similar arguments values.



(a) region growing using thr: 50

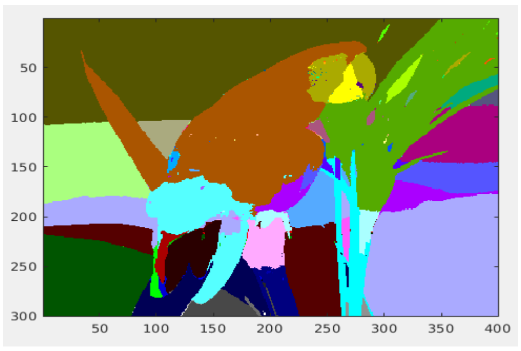


(b) region growing using thr: 75

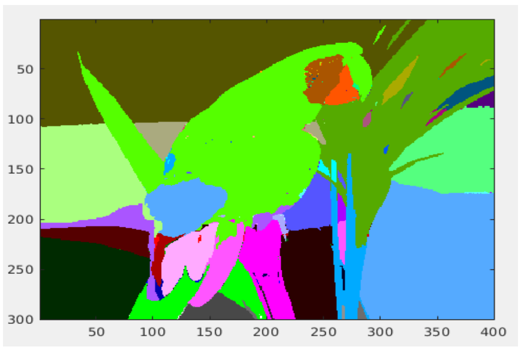


(c) region growing using thr: 100

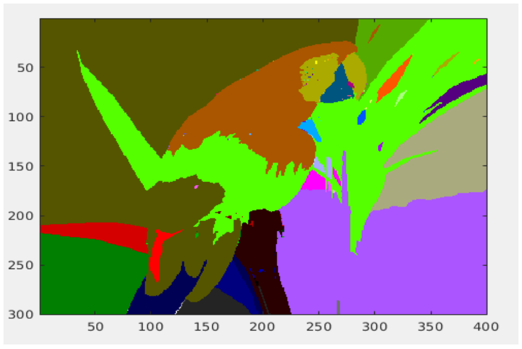
Figure 13: Region growing threshold comparison



(a) region growing using thr: 50



(b) region growing using thr: 75



(c) region growing using thr: 100

Figure 14: Region growing after mean shift sthreshold comparison

As we can see, the regions are more consistent to the threshold value if we use it after a mean shift that doesn't need to be very restrict. The colors change because as we said, region growing uses a tag to identify the pixel region and destroy de color information, but the shape and number of regions is more robust in Figure 14 than in Figure 13

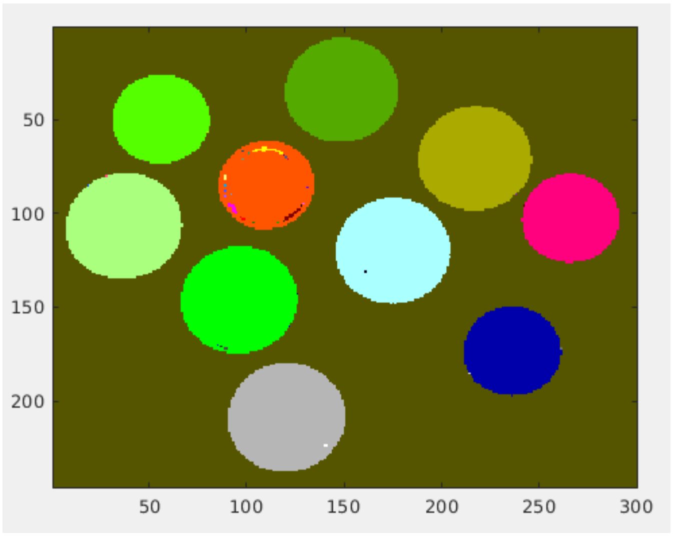
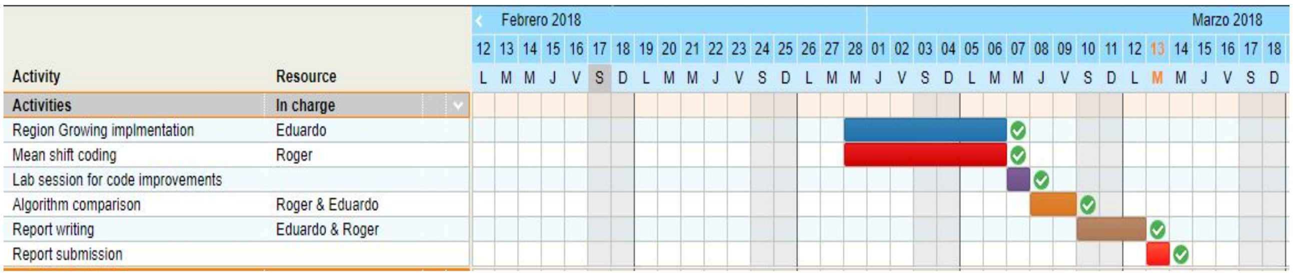


Figure 15: Coins segmentation using mean shift + region growing

In Figure 15 mean shift and region growing have been also combined to segment the coins. As we can see, there is still some small regions but the result looks much more clear than other results as in Figure 5 for example, and without tuning any parameter.

6 Project management

In Figure 16 a Gantt diagram is presented to picture the division of tasks and the amount of time that was dedicated to the lab development.



		Febrero 2018																	Marzo 2018																	
		12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18
Activity	Resource	L	M	M	J	V	S	D	L	M	M	J	V	S	D	L	M	M	J	V	S	D	L	M	M	J	V	S	D	L	M	M	J	V	S	D
Activities	In charge																																			
Region Growing implmentation	Eduardo																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	✓											
Mean shift coding	Roger																	🟥	🟥	🟥	🟥	🟥	🟥	🟥	✓											
Lab session for code improvements																									🟪	✓										
Algorithm comparison	Roger & Eduardo																									🟧	🟧	✓								
Report writing	Eduardo & Roger																											🟫	🟫	🟫	✓					
Report submission																															🟥	✓				

Figure 16: Gantt diagram of the task planning

7 Conclusions

In this lab we have been studied and implemented two different segmentation methods based on regions, region growing and mean shift. The region growing algorithm provides a fastest implementation than the mean shift, but it doesn't preserve any color information and its result is highly sensitive to the threshold value, and also it depends on where the seeds are placed and how many of them there are.

On the other hand, mean shift it takes much more time but the clustering result is more robust and it preserve some color information. Regions are defined as clusters that are formed by a subset of points based on its position on the feature space.

We can use region growing with the output of the mean shift to reduce the sensitivity of the color threshold value since the mean shift shrink the clusters and make them more different and robust to noise.

8 How to run it

To run the code, we provide 4 different files. Two of them are the functions, which doesn't need to be touched or modified, "region_growing.m" and "meanShiftPixClust.m". Also, two scripts are provided, to make it easier to test. "lab1.m" has all the steps needed to run the region growing algorithm and plot the results. On the top of the script you will find the 3 parameters you need to modify: the name of the image file, the threshold and the connectivity (4 or 8).

The script named "test_meanshift.m" will run the mean shift algorithm, plot the result over different iterations and the final output vs the original one, even showing the clusters in the feature space. On line 5 you can modify the name of the image file and in line 7 you can modify the parameters of the mean shift algorithm as the kernel or the threshold, but we only recommend to touch the threshold.

References

[1] Iman Avazpour, M Iqbal Saripan, Abdul Jalil Nordin, and Rajaa Syamsul Azmir Iman Abdullah. Segmentation of extrapulmonary tuberculosis infection using modified automatic seeded region growing. *Biological procedures online*, 11(1):241, 2009.

[2] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.