Implementation of Region Growing Algorithm

Rodrigo Daudt, Songyou Peng

I. INTRODUCTION

This report describes an implementation in MATLAB of the region growing algorithm for unsupervised image segmentation, which was developed for the Scene Segmentation and Interpretation module at University of Girona.

II. ALGORITHM ANALYSIS

Region growing is a region-based algorithm whose main idea, as the name suggests, is to grow regions from starting points (seeds) based on a predefined rule until all pixels have been assigned a label. We also see the method as a pixel-based algorithm, because the initial growing points play influence the final result [1].

First of all, the algorithm starts growing an initial region from a starting point [2], which can be given by the user or automatically generated. We will see in the section V that different starting points may lead to different results. After acquiring the starting point for a given region, all its neighbors are taken into account and analysed based on an aggregation criteria which will define whether or not the pixel should be added to the current region. If the analysed pixel meets the criteria, we can say it belongs to the current region so the same label is assigned to this new pixel. We then analyse the neighbors of this newly added pixels and continue this process recursively until no more pixels fit the aggregation criteria.

When all the neighboring pixels to the current region have been analysed and not added to it, the current region has grown completely and another unlabeled point is chosen as the new starting point for a new region. Finally, when all points in the image have been labeled, the algorithm stops and we acquire the segmentation result.

In our case, we consider all 8 neighbors of the center point and our aggregation criteria is

$$\Sigma_z |I_z(x,y) - \mu_{R_i}| <= \Delta$$

where z is number of channels, that is, $z = 1$ for a grayscale image and $z = 3$ for the color image. $I_z(x, y)$ is the value of the current point in layer z. We consider the RGB channels for color images. μ_{R_i} is the mean of region R_i so far and Δ is a threshold given by the user and is used as a tuning parameter.

III. DESIGN AND IMPLEMENTATION

The region growing algorithm described in Section II was implemented in a MATLAB function for easy application to any amount of images. The function has the form

$$[final] = rg(image, delta\_in)$$

where $image$ is the input image which can be either in grayscale or in color, $delta\_in$ is the threshold used on the aggregation criteria, and $final$ is the output image where the labels for each pixel are stored. The output of this function is an image with only one layer regardless of the nature of the input image, given the labels are simple numeric values. Since our function can be applied to images with more than one layers, the value of $delta\_in$ is multiplied by the number of layers in the image so that a given threshold value has roughly the same results when applied to color images or grayscale images. The seeds for each of the regions was obtained by a simple sweep through all the pixels of the image.

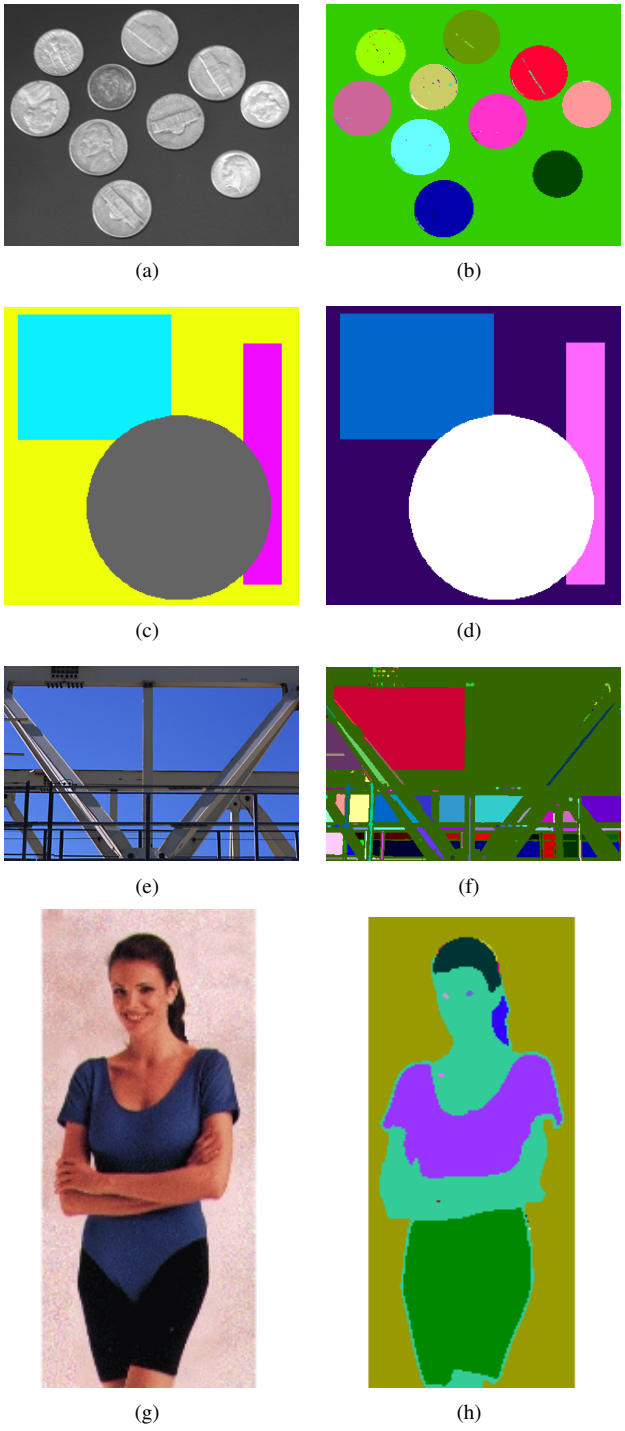
The algorithm was implemented in a way so that the regions grew in steps called generations. Since the order of the neighbours to be analysed is not defined in the region growing algorithm, it is assumed that for a given state of the region all neighbours can be compared without updating the region statistics for every added pixel. In other words, at a given state of the region, since any pixel can be the next one to be analysed, every pixel can be correctly analysed using the current statistics.

The algorithm worked in the following way: at a given step, all the neighbours from the pixels added in the previous generation are analysed and, if appropriate, are added to the region. When a pixel is added to the region, its index is added to a list to be processed later on. But the generational approach avoids having one constantly resized queue, which can slow down the execution for poorly implemented queues, and also significantly reduces the number of times that the mean values of the region are calculated. This process also tends to resize smaller arrays (used here as queues) instead of the complete queues for all pixels to be processed, which can be faster depending on the implementation of MATLAB, and pixels are deleted from the queue in batches which also makes the program faster. Once the queue of pixels added in the previous generation has finished its processing, the queue of newly added pixels (called current generation) is transferred to the previous generation queue and a clean queue starts being written. At this moment, the means for each of the channels of the pixels in the regions is recalculated.

For finding the neighbours of a pixel, all the offsets for each of the possible neighbours is stored in an $Nx2$ matrix ($8x2$ for 8-neighbourhood) to avoid hardcoding the neighbourhood operations and copying and pasting the code multiple times. With this approach, any different neighbourhood configuration can be used by simply changing the offset matrix.

IV. RESULTS

A. Segmentation results



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 1. Segmentation results

B. Quality and speed

When the thresholds are chosen wisely, the qualities of the "coin" and "color" synthetic images' segmentation results are good. However, for the "gantrycrane" and "woman", choosing an appropriate value for the threshold is not enough. Due to the fact that the structure of "gantrycrane" is complicated and "woman" has a texturised background, we need to preprocess the images. Here we apply 5×5 Gaussian filters on them in order to eliminate some small regions in the results, which can significantly improve their quality.

As for the speed, in the environment Mac OS X 10.10.5, 2,7 GHz Intel Core i5, 8 GB 1867 MHz DDR3, "coin" and "color" take around 0.7s and 0.8s to be segmented, while "gantrycrane" and "woman" take around 1.7s and 0.3s.

V. DISCUSSION

During the process of implementation of region growing method, we had some interesting observations about the segmentation results. In this section we will first talk about different results due to the change of some parameters, and then compare the method with Fuzzy C-means method.

A. Different thresholds

The threshold of aggregation criteria is a parameter which may lead to significant changes.

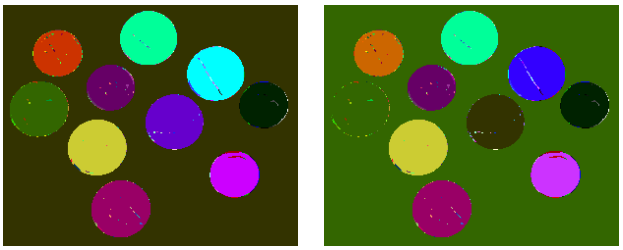
(a) $\Delta = 5$ (b) $\Delta = 50$ (c) $\Delta = 100$

Fig. 2. Segmentation results of different thresholds

The image in Fig. 2(a) is segmented to a large number of fragments (over segmentation), while too many details of the woman merge together in 2(c) (under segmentation). The properly segmented result is shown in Fig 2(b) when $\Delta = 50$.

B. Different starting points

We found out that, if different starting points are chosen, the segmentation results may differ from one to the other. In Fig 3, it is easy to notice that there is a coin that is grouped as the background.



(a) (1,1) (b) Middle (123,150)

Fig. 3. Segmentation results of different starting points

The explanation for this situation is that when the method starts from different growing points, the growing of the

region may affect the mean of the region in different ways, changing the result. Especially at the beginning of the growing, if a pixel with a relatively high value is added, the mean of the region may change a lot, and then some pixels which should not be in the region may be grouped into it. As a result, the final results may be quite different.

C. Sensitive to noise

One main drawback of region growing method is its sensitivity to noise and texture. When the input image contains some noise, the method is likely to have many small regions in and around the noise. For example, in Fig 4(a) what we can notice is that, there are many small regions around the boundary of the woman. If applying a 3×3 Gaussian filter before segmenting, the result is quite decent.

(a) No smoothing (b) Smoothing

Fig. 4. Segmentation results between the original image with and without smoothing

D. Comparison with Fuzzy C-means

For comparison, the Fuzzy C-Means clustering algorithm was applied to the same set of test images. One main difference between these two algorithms is, as would be between any region based method and a clustering method, that the region growing algorithm takes into account the position of the pixels in the image when creating the regions, while the FCM algorithm is capable of generating discontiguous regions.

The results of the FCM algorithm can be seen in Figs. 5-8. The FCM implementation in MATLAB takes as an input parameter the number of regions we want to segment the image in. The first try was to segment the coins image in 11 regions, being one for each of the 10 coins and one for the background. This did not work properly, since the FCM is not able to detect the boundaries between the objects as the region growing algorithm is. On the other hand, when trying to apply FCM to create two clusters (one for coins and one for background) the results are very good, as can be seen in Fig. 5. The downside of this condition is that FCM is not able to separate the coins in different regions.

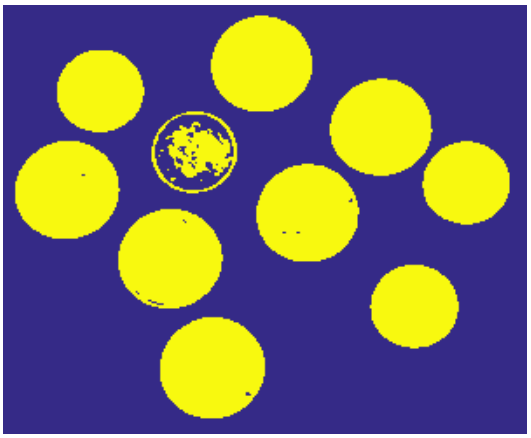


Fig. 5. Result of the Fuzzy C-Means clustering algorithm - coins

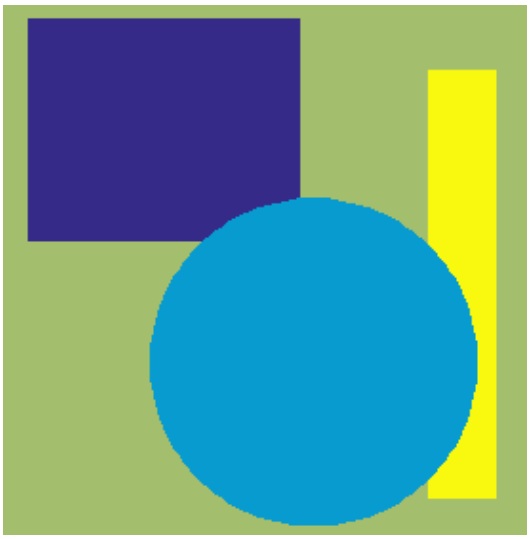


Fig. 6. Result of the Fuzzy C-Means clustering algorithm - color

When applied to the colors image, the FCM algorithm either achieves complete success or complete failure. This seems to happen due to a poor initialization of the clusters, and may be amplified by the strong homogeneity of the pixels in the image. The result of a successful run is shown in Fig. 6.

When applied to the gantrycrane and woman images, the FCM performed better than the region growing algorithm even with no preprocessing. These results can be found in Figs. 7 and 8 respectively. On the case of the gantrycrane image, the success of this algorithm in comparison to the region growing algorithm can be explained by the fact that since the position of the pixels is not taken into account for the FCM, all the multiply divided regions can be easily put back together since their pixels have similar values. For the woman image, the FCM seems to be more robust to the texturized background. One possible explanation for this is that FCM forces a low number of clusters, therefore the oversegmentation of the background is kept under control.

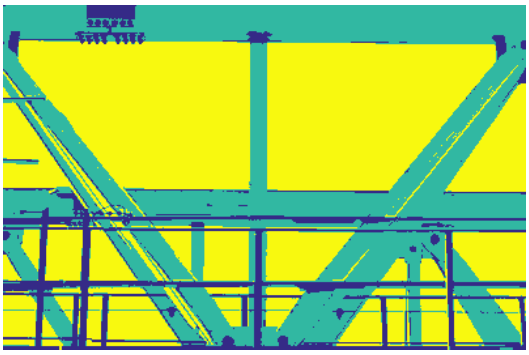


Fig. 7. Result of the Fuzzy C-Means clustering algorithm - gantrycrane

Fig. 8. Result of the Fuzzy C-Means clustering algorithm - woman

VI. PROJECT MANAGEMENT

It was decided at the beginning of the execution of this project that, for learning reasons, each of the members would develop their own implementations of the region growing algorithms. After two working versions existed, features of both versions were combined in order to produce a final version of the region growing function. The Github platform was used as a collaborative tool for the code development[3], and the report was written using the online LaTeX platform Overleaf.

The rest of the project, including testing, image acquisition, application of the Fuzzy C-means algorithm, and writing of the report, was done with no clear separation of tasks between the members.

VII. CONCLUSIONS

In this lab, we implemented the region growing method to segment images and acquired accurate results in a short time. We analyzed the influence of different starting points and grouping thresholds on the results. Also, the method can not provide a good result for noisy images or images with complicated structure without some kind of preprocessing.

We also compare region growing method with Fuzzy C-Means clustering, and observed that the clustering technique seems to have similar or worse performance on simple images, but it performs significantly better on spatially complex images.

REFERENCES

[1] Wikipedia, The Free Encyclopedia, s.v. "Region growing," (accessed February 29, 2016), https://en.wikipedia.org/wiki/Region_growing.

[2] Adams, Rolf, and Leanne Bischof. "Seeded region growing." Pattern Analysis and Machine Intelligence, IEEE Transactions on 16, no. 6 (1994): 641-647.

[3] Rodrigo Daudt & Songyou Peng, SSI-Lab1, (2016), GitHub repository, https://github.com/rcdaudt/SSI-Lab1.