UNIVERSITAT DE GIRONA

SCENE SEGMENTATION AND INTERPRETATION

LABORATORY REPORT

Image Segmentation

Author
Mohit VAISHNAV
Malav BATERIWALA

Supervisor
Dr. Xavier LLADÓ

March 13, 2018

1 Introduction

Image segmentation is an important process in the field of computer vision which corresponds object into different parts using various operations like color, texture or any other segmentation method. This data can be used for many operation such as classification of objects, detection of an object or just 3D reconstruction. Basically segmentation divides pixel clusters into regions using various methods and those clusters information can be used to analyze image to gather more data. So for every pixel in any image , we can segment the pixels according to the catergory having similar characteristics or by evaluating the neighborhood pixels and categorizing them.

Image Segmentation of images is one the most difficult task considering the part of image processing, but in return it also has a broad range of application in many diverse fields of engineering and research. Segmentation accuracy determines the success of the algorithm or the task we want to perform later is a success or a failure. In this lab work, we will implement on the algorithm of region growing which is one of the basic algorithm that processes digital image into different partition and then analyze the design and implementation of it. Then we will compare this algorithm with other image segmentation algorithms to check the comparison between the different algorithms.

2 Implementation

There are two files in the folder namely, *main.m* and *regiongrowing.m*. To make the program fuctionality robust we have altered the code in such a way that given a set of images inside *p1_images* in *.jpg* format, it will save all the region labelled images in folder named *seg* kept inside the main folder in the same format.

Other parameters which could be altered are such as *threshold* and *neighbourhood*. By default the *neighbourhood* values are taken as 8. Nomenclature of the saved images are very simple, they are named in the format mentioned below:

thresholdvalue_neighbours_OriginalImage.jpg

To characterize the various performance measurements we have created a *.txt* file with the name *changing.txt*. It saves the following information: *name, number of regions, connectivity, threshold and time taken* for the set of images contained in the folder *seg*.

Now coming to the Algorithmic part of our program; we have passed on parameters such as *Image, Threshold and Neighbourhood* to the *Regiongrowing.m* file. There we have created an image of the same dimension as that of original one with zero values which is an indicator to stop if all the zeros of the images have been replaced with some value signifying the region number.

Next we have defined the neighbourhood of 8 nearby pixels which according to the user input considers just 4 if it has been defined. We start with initialization of *Number of Regions* as 0 value which increments if the desired condition is fulfilled with in the structure. Again to optimize the code, we have created an array to tackle the dimensionality issue where we have defined average parameter (*avg_par*) which computes the mean of the region as it grows as 1-D with either one value or three based on the input image. Even to compute the distance which is to be compared with the *threshold* value we have used *euclidean distance*. As soon as a pixel is visited its value in reference image, *Image_segmented* changes to some number and this is not repeated again which is taken care before entering any loop.

Output of the region growing function file are the number of regions and the segmented image corresponding to the input ones.

For a comparison purpose we have implemented the *Fuzzy* means algorithm whose number of cluster centers are given by the variable $fuz\_cluster$. Input image is passed in *double* format with $fuz\_cluster$ to get the desired result. The output is shown using *imagesc* MATlab command to have a better understanding because of colored representation. At the end we saved the output in the *.jpg* format with the following name convention:

$$name = fuzzy\_Numberofclusters\_imagename$$

3 To run the code

All the implementation has been done in the *.m* file where the parameters could be changed according to the desired output. We have created a program to run it on all the images contained in the folder *seg*. In this *MATLab* reads all the *.jpg* file and limit the loop end accordingly. Name is extracted from this structure which is used in appending at the end of the output file along with other variables. For saving the *regiongrowing* output image, we put together *connectivity* and *threshold*. Option to display figure has been turned off because of accumulation of large number of images being displayed all together. Above part of the code contains the program related

to regiongrowing whereas bottom part is related to k-means. You can comment one section while running the other so that there is no error message is generated because of non declaration of variables.

To study about the complexity and run-time of the code, we created 2 files, one for each, region-growing and fuzzy. Each line clearly mentions about the various parameters used along with the time taken for each individual image. To write into the file, *a* is used which appends the information along with the others already computed.

4 Results

These are the results obtained when we tried with many different parameters and images.

All the results are with different threshold and type of neighbourhood system used, and also they are compared with fuzzy logic segmentation.

Sr.No.	Name of Image	Type	Size (in pixels)	Conectivity	Time taken for threshold and Regions		
					T= 20 and R	T=50 and R	T=100 and R
1	Coins	Gray	472x371	8	16.42 and 3474	21.08 and 901	44.27 and 197
2	Color	RGB	447x373	8	16.33 and 3272	27.15 and 199	36.70 and 92
3	GantryCane	RGB	572x360	8	26.45 and 6473	33.34 and 1428	45.88 and 640
4	Woman	RGB	388x356	8	9.6 and 2150	12.75 and 464	23.3 and 234
5	Coins	Gray	472x371	4	25.4 and 5137	19.7 and 1546	18.6 and 256
6	Color	RGB	447x373	4	22.3 and 1645	20.6 and 391	20 and 102
7	GantryCane	RGB	572x360	4	29.5 and 8995	29.68 and 2621	27.5 and 863
8	Woman	RGB	388x356	4	12.4 and 2872	11.9 and 608	15.3 and 332

Figure 1: Time taken by Images provided

Sr.No.	Name of Image	Type	Size (in pixels)	Time taken by no. of Clusturs		
				F = 5	F = 7	F = 10
1	Coins	Gray	472x371	12.9	18.63	30.25
2	Color	RGB	447x373	7.78	20.37	31.32
3	GantryCane	RGB	572x360	9.7	25.2	38.5
4	Woman	RGB	388x356	9.13	12.24	19.23

Figure 2: Time taken by Fuzzy Segmentation

5 Difficulties

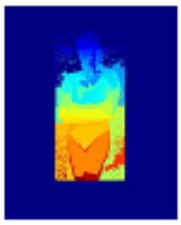
Implementation of this algorithm could be done in various ways which either could take seconds to run or minutes else in worst case hours together. We had the *Intel Core 2 Duo* processor and *Core i3* which does not even supports parallel processing, hence had to optimize the code to the maximum level which we did. Although we found a bottle neck in the code where we had to save the list saving the location of the pixel visited and move accordingly and was of importance. It was consuming more than 60% of the time of which we came to know with the help of our supervisor. It was also a learning for both of us as we have never encountered a situation where we ever looked for bottleneck issue of any code and work on that part accordingly.

Our next issue was to look for the optimum threshold value which could be generalized for any kind of image. To have the feeling of *oversampling, undersampling and optimum sampling* we tried executing our code for various images including many others from what have been provided. We came to know that there cannot be any one particular threshold for all kind of images. It depends on various factors like texture, amount of information in the image, lighting effect, patterns of the image etc. We found that using *threshold* value as 50 we were able to get a decent segmentation for most of our image dataset.

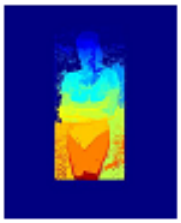
6 Conclusion

We have observed various trends while playing with various parameters around the code. As we increase the connectivity from 4 to 8, the time taken is decreased which seems logical. But this trend is not applicable as we increases the *threshold* to a higher value. Based on our analysis we would like to propose a *threshold* of 50 and *connectivity* of 4. Results have been compared with $Fuzzy - C$ means algorithm where we tried changing the number of clusters in a range of 5,7 and 10. Tradeoff between time taken and the number of clusters made us to draw a conclusion that we should take the $cluster$ as 7 which takes almost same time as that of *regiongrowing* with *threshold* 50. Regional growing is the pixel by pixel implementation whereas clustering based method has high positional relation and considers the distance between between data. Looking at the results given by us we can say that there is no particular best method, it all depends on user input and its a tradeoff between both methods.

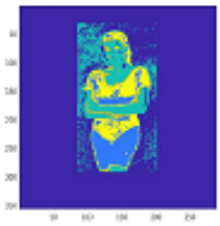
ORIGINAL IMAGE



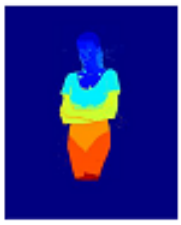
N=4 T=20



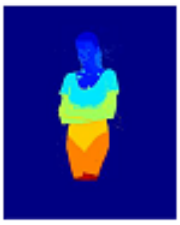
N=8 T=20



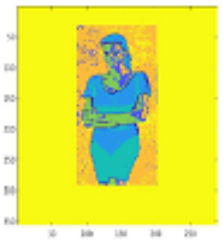
F = 5



N=4 T=50



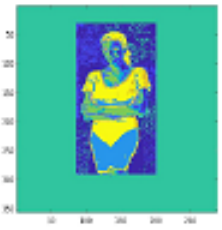
N=8 T=50



F = 7

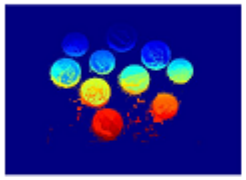
N=4 T=100

N=8 T=100

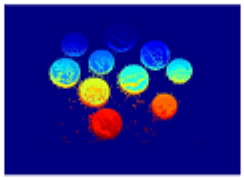


F = 10

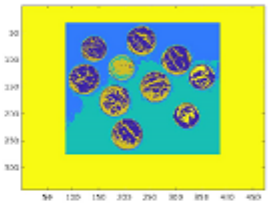
ORIGINAL IMAGE



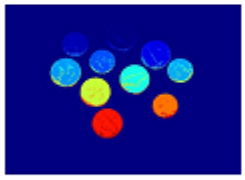
N=4 T=20



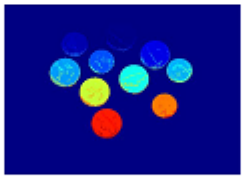
N=8 T=20



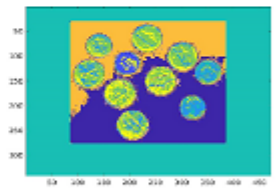
F = 5



N=4 T=50



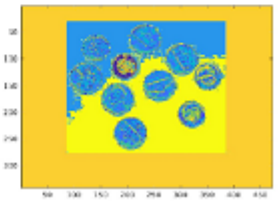
N=8 T=50



F = 7

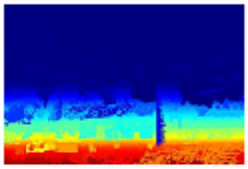
N=4 T=100

N=8 T=100

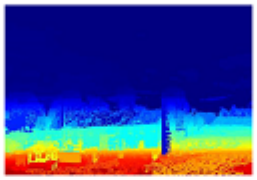


F = 10

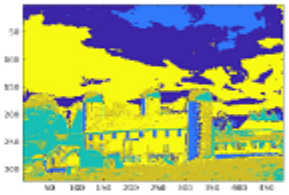
ORIGINAL IMAGE



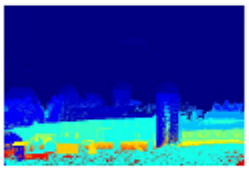
N=4 T=20



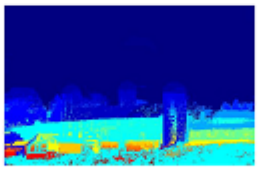
N=8 T=20



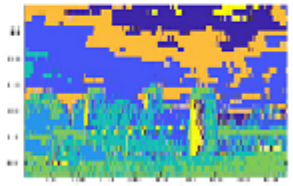
F = 5



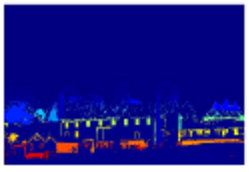
N=4 T=50



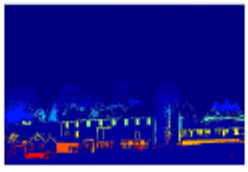
N=8 T=50



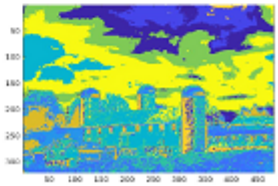
F = 7



N=4 T=100



N=8 T=100



F =10