Scene Segmentation and Interpretation
Image Segmentation - Region Growing

SAVINIEN BONHEUR & ALBERT CLÉRIGUES
Universitat de Girona
March 2, 2017

1 Introduction and problem definition

The goal of this coursework is to introduce us to the image segmentation problem by developing and implementing the Region Growing algorithm and comparing it against more advanced image segmentation algorithms.

Image segmentation is often defined as the problem of localizing regions of an image relative to content, which is inherently a subjective and domain-dependant problem since having a good segmentation requires defining what the content is.

2 Algorithm analysis

The region growing algorithm is a region-based image segmentation algorithm that works at pixel level. The operation is simple, initialize a new region by picking an unlabelled pixel and repeatedly explore the neighbouring pixels of the region, appending them if they comply with the aggregation criteria. When there are no more neighbours to add to the current region, start another region by picking an unlabeled seed pixel. Iterating in this way until all pixels in the image belong to some region as seen in Algorithm 1.

3 Design and implementation of proposed solution

The implementation has been done using the Matlab programming language with a sequential approach, meaning it selects the new region seeds by locating the next row-wise sequentially unlabelled pixel. The exploration of the neighbours is done with Breadth-first search, and uses a system of two queues, one for the current level and other that will hold the neighbours of the next level to be explored.

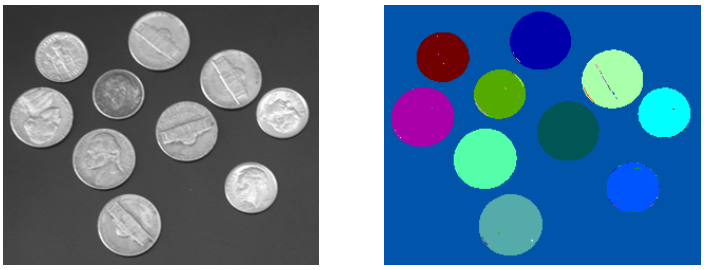
Algorithm 1: Pseudo-code for our growcut algorithm implementation

```
1  function growcutsegmentation (image, threshold, connectivity);
2  n ← 0
3  while ∃ unlabeled pixels do
4      n ← n + 1
5      R_n ← First sequentially unlabeled pixel      // Init region n
6      S_n ← ∅          // Initialize statistics region n+1
7
8      Q_next ← Q_next ∪ R_n
9      while Q_next ≠ ∅ do
10         Q_current ← Q_next
11         foreach node n ∈ Q_current do
12             foreach neighbour neigh ∈ connectivity do
13                 if neigh complies aggregation criteria then
14                     S_n ← neigh
15                     R_n ← R_n ∪ neigh
16                     Q_next ← Q_next ∪ neigh
17                 end
18             end
19         end
20     end
21 end
```

4 Experiments and results

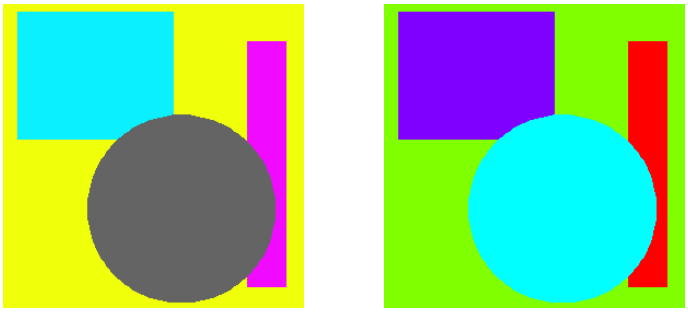
4.1 Asserting correct implementation

First we developed a series of tests to assert the core functionality of our implementation such as grayscale segmentation, color segmentation and different connectivities.



(a) Original (b) Segmented thresh=65

Figure 1: Basic test with grayscale image



(a) Original (b) Segmented thresh=1

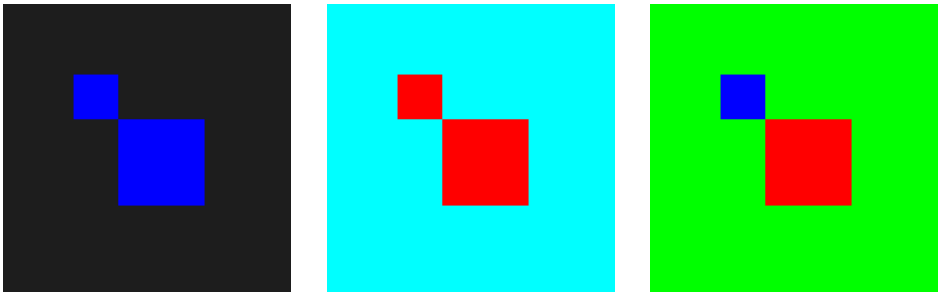
Figure 2: Basic color segmentation test

We also designed a synthetic test to unambiguously assert correct color segmentation by providing a color image whose first channel and grayscale version where a uniform color.

(a) Original (b) Original grayscale (c) Segmented thresh=1

Figure 3: Synthetic color segmentation test

Finally we designed a last test to assert correct connectivity handling:



(a) Original (b) 8 connectivity (c) 4 connectivity

Figure 4: Connectivity test

4.2 Execution time

We segmented the four given images with our algorithm measuring the execution time:

Filename	Threshold	Execution time
color.tif	1	0,72 s.
coins.png	65	0,59 s.
woman.tif	85	0,31 s.
gantrycrane.png	120	1,07 s.

We then tested on synthetic images how our implementation complexity scales with increasing image size and also for the number of regions in an image:

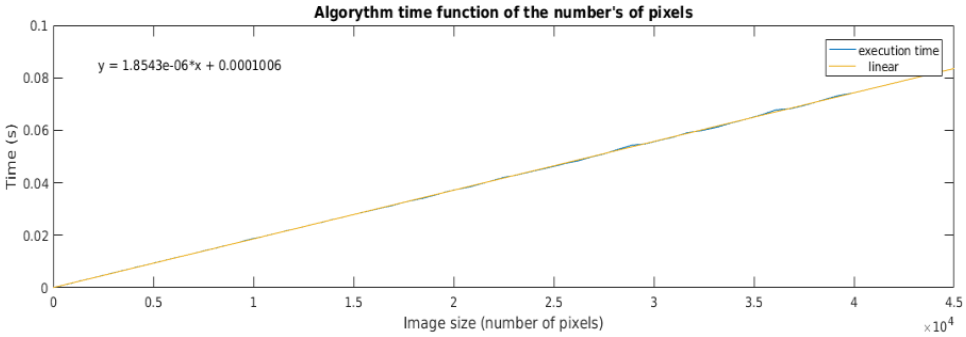


Figure 5: Execution time vs. number of pixels in image

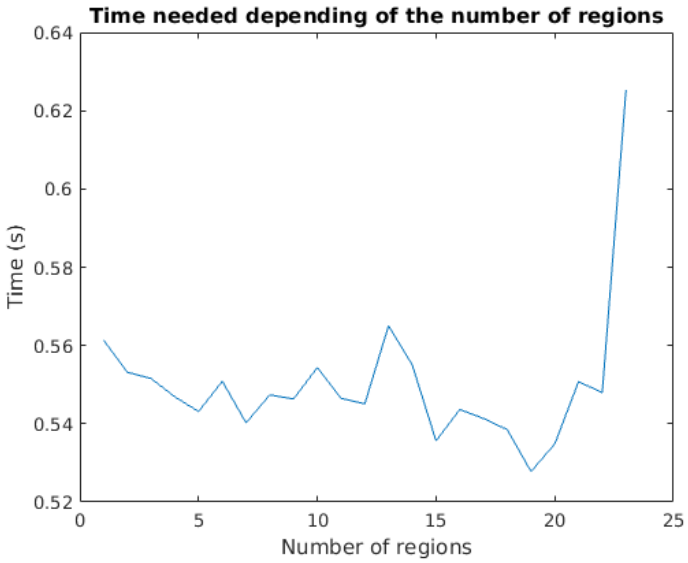


Figure 6: Execution time vs. number of regions

We conclude that implementation is well done since it linearly scales with increasing image size at a rate of $2\mu s.$ per pixel. The number of regions on the synthetic image don't change significantly the execution time.

5 Region Growing vs. Fuzzy C-means

In this section we compare our implementation of region growing with the Fuzzy C-means from Matlab. The main difference between them is that region growing is region-based, ensuring that each label is a connected region, while FCM is a clustering based method whose output regions could not be spatially connected.

5.1 Input parameters

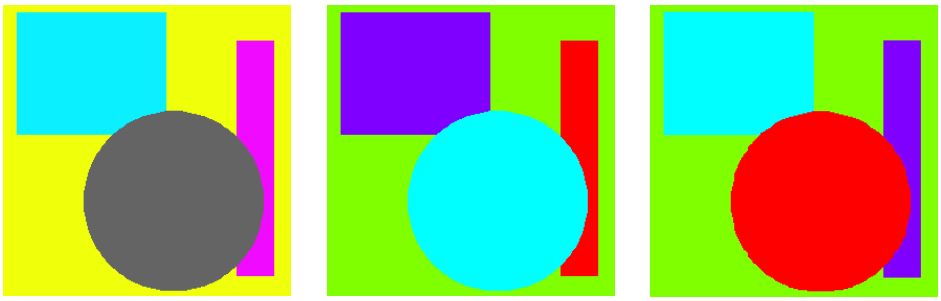
The input parameters for both algorithms are summarised in Table 1:

Region Growing:	threshold	connectivity
Fuzzy C-means:	num. clusters	num. overlaping clusters

Table 1: Input parameters for Region Growing and FCM algorithms

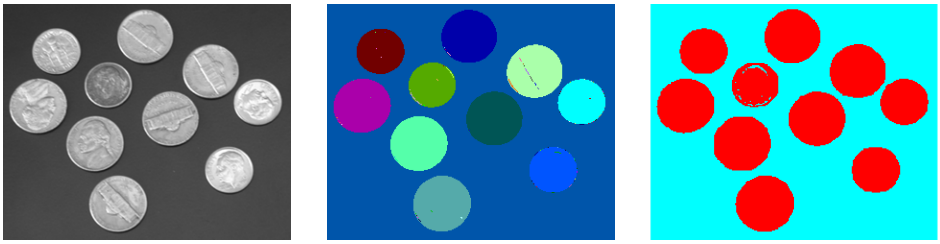
As we can see FCM requires an a priori estimation of the number of clusters in the image and how much should they be allowed to overlap. This is a big disadvantage against region growing for general segmentation, since it doesn't require any previous knowledge of the contents of the image.

5.2 Segmentation Comparison



(a) Original (b) RG (c) FCM

Figure 7: RG and FCM segmentation in *color.tif*

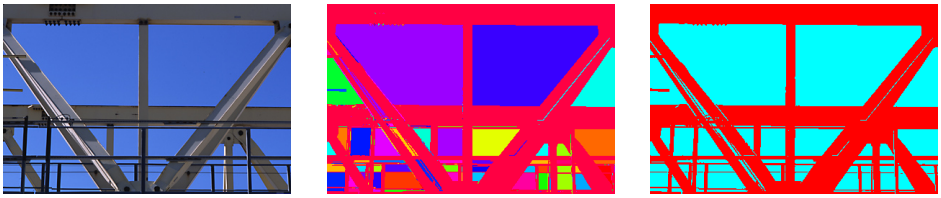


(a) Original (b) RG (c) FCM

Figure 8: RG and FCM segmentation in *coins.png*

(a) Original (b) RG (c) FCM

Figure 9: RG and FCM segmentation in *woman.tif*



(a) Original (b) RG (c) FCM

Figure 10: RG and FCM segmentation in *gantrycrane.png*

According to the obtained results we conclude that FCM works better than Region Growing for images where we know the number of color clusters related to its content. However, if we do not have this information the usage of FCM will surely lead to under or oversegmentation. For general segmentation or connected components using the region grow algorithm would lead to more accurate results, and posterior clustering could always be performed on the regions themselves.

When faced with natural noisy images both agorithms have problems segmenting the fuzzy areas as seen in Figure 9. The use of a small blurring filter would help with the multiple single-pixel regions that appear in Region Growing and with the noisy borders of FCM.

We conclude that both algorithms deliver similar results, being the main differences the a priori information about the image and the region or clustering based results.

6 Organization and development of the coursework

The workload estimation and distribution finally was as follows:

Tasks	**Time**	**In charge**
Basic algorithm implementation.	4h.	Albert & Sav
Add color aggregation criteria.	2h.	Savinien
Design and testing of Region Growing.	2h.	Savinien
Optimize execution time.	2h.	Albert
Implementation of FCM and FCM vs RG tests.	4h.	Savinien
Write coursework's report.	8h.	Albert & Sav

Table 2: Workload time estimation and distribution

7 Conclusions

After the implementation and testing of the region growing algorithm we can draw several conclusions:

- Although being a simple and basic segmentation algorithm it can provide quite good segmentations depending on the type of image.
- The algorithm is really sensitive to noise, producing many small regions on high frequency areas without connectivity.
- The only needed free parameter is the threshold for the aggregation criteria, which can be easily adjusted with supervision.

Region growing is fitted for images with connected regions without many high frequency changes, such as the *coins.png* image for example. In any case it would be positive to use a low pass filter with a small-sized kernel such as average or gaussian to improve the sloppy connectivity in high frequency areas.