Advanced Image Processing

Coursework 2

Image Segmentation

Region Growing

Alper Akcoltekin

Abdoulaye Diakité

22.02.2011

INDEX

1. Introduction **3**

2. Algorithm Analysis **3**

3. Design and Implementation **4**

4. Experimental Results and Results Analysis **6**

5. Organization and Development of Coursework **10**

6. Conclusion **10**

References **10**

1) Introduction and Problem Statement

Image segmentation is a popular branch in the field of computer vision. Main goal in segmenting an image is to seperate it into non-overlapped regions such that the union of all the regions is equal to the entire image. It is basically used to locate objects, boundaries, lines, curves etc. in an image and it is useful in many domains such as medical imaging, satellite imaging, face/fingerprint recognition, machine vision and so on.

There are many methods to segment an image, but for this coursework we will use the well-known Region Growing algorithm. This algorithm is pretty fast and it gives satisfactory results. After segmenting an image, we find the DSC (Dice Similarity Coefficient) comparing the ground truth (actual result) and the result of our algorithm. This ratio indicates how well an image is segmented.

2) Algorithm Analysis

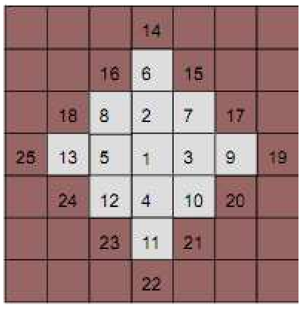
Region Growing algorithm is a seeded growing algorithm. A set of seeds are placed on the image – on the parts/objects to be segmented – and each seed starts to create a region by exploring their neighbors. The explored pixels are taken into the region if they are similar to that seed; otherwise, they become a seed and they grow their own region. Exploring continues recursively, i.e. seed explores its neighbors, then its neighbors explore their own neighbors, and so on. Similarity is the distance between the intensity value of a pixel and the mean intensity value of the pixels in that region. This distance is limited by a threshold which may vary for each image. Changing the threshold increases or decreases the sensitivity when placing a pixel into a region, so it should be chosen carefully and precisely.

The most important issue of this algorithm is the number and placement of seeds. Manually placing seeds takes time and it is not a good way when dealing with complex images. In order to automatize the seed placement process, we generate a specific number of seeds and place them randomly on the image.

3) Design and Implementation

Region Growing algorithm is actually a recursive algorithm, but since we use MATLAB and since it doesn't allow recursion, we implemented it as a sequential algorithm.

Exploring process was implemented using a *queue*. When a pixel explores its neighbors, it finds the ones that are similar to itself and puts them in a queue. The neighborhood can be either 4 or 8; but considering the execution time, we used 4 neighborhood in our implementation. The same process is applied for the pixels in the queue, i.e. every pixel explores the similar neighbors and puts them in the queue. When the queue is empty – in other words, when the region reaches the maximum area – this seed has grown as much as possible; so we start to grow another seed. We have a *visited matrix* to know which pixel is explored. The figure below illustrates the order of the pixels put in the queue.



Region Growing using a queue

In order to label the pixels with regions, we keep a *region matrix* – which will actually give us the segmented image in terms of region numbers – and update it as the seed explores its neighbors and grows its region. For example: if a neighbor is similar to that seed, it is labelled with the number of that region in the region matrix.

We generate random seeds, so some seeds may be left out or not given a region if all of its neighbors are visited. For those pixels, we estimate the probable region that it might take place. Looking at the regions of the neighbors, we find the most frequent region and put the seed into that region. For example; the seed below will be labeled as region 1.

1	1	1
1	Seed	1
2	2	1

Here is the psuedo-code of our algorithm:

```
Initialize the region and visited matrices as zero
Generate the seeds and create a seed list
currentRegion = 1

WHILE seedList ≠ ∅
    pick the first seed S;
    IF explored(S)
        ignore S and skip this loop;
    region = region ∪ S;
    visited(S) = true;
    regionMatrix(S) = currentRegion;
    initialize the queue Q = { };

(1) FOR each neighbor $N_i$ of S

        IF visited($N_i$) = false
            visited($N_i$) = true;
            difference = $\left| intensity(Ni) - mean(region) \right|$;
            IF difference < Δ
                region = region ∪ $N_i$;
                regionMatrix($N_i$) = currentRegion;
                Q = Q ∪ $N_i$;
            ELSE
                Put $N_i$ in the seedList as a new seed;
            END IF
        END IF
    END FOR

    IF all the neighbors were visited
        Put the seed in probable region estimated using neighbor regions;

    WHILE Q ≠ ∅
        Pick the first pixel P from Q;
        Apply (1) to neighbors of P;
    END WHILE

    increment currentRegion;

END WHILE
```

After applying the algorithm, we apply *color coding* to the final regionMatrix and assign randomized colors to each region. Since it's a random process, colors may not be excellent all the time. After we obtain a segmented color image, we convert it to a *binary image* and compare with the ground truth. According to some metrics (true/false negative/positive) we find the DSC.

4) Experimental Results and Results Analysis

We tested our algorithm on both grayscale and color images. When testing our algorithm, there were several parameters that varied: explored neighborhood, threshold values and seed number.

Explored neighborhood can be either 4 or 8, but since it requires more time to process 8 neighbors than 4; we mostly worked 4-neighborhood and however, the results were better.

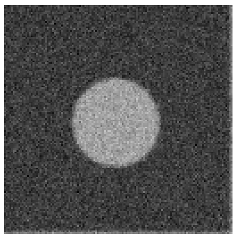
Threshold values are very important in results; so we could only find the optimal threshold values after testing multiple times. Since each image has a different characteristic and complexity, threshold values vary for each image.

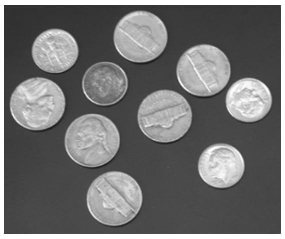
Seed number is fixed as 100 in our tests; because more seeds caused in poor quality results and less seeds caused insufficient results. This seed number gave the best solution for all tested images.

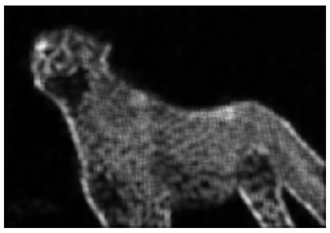
Furthermore, we tested the algorithm on the results that we obtained from Coursework 1. We segmented the textured image "feli".

4.1) Grayscale Images

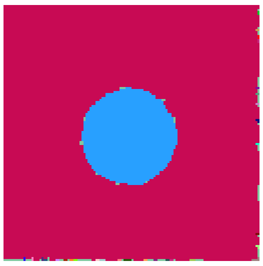
Here are a set of original grayscale images before applying the algorithm:

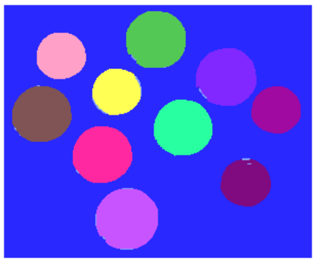

gris21.tif

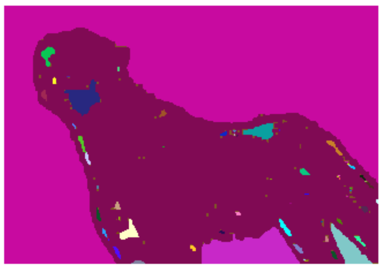

coins.png


feli.tif

Here are the results after applying Region Growing and color coding:

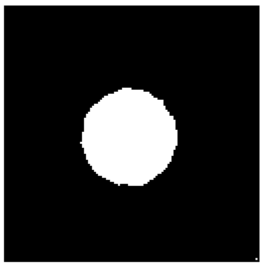


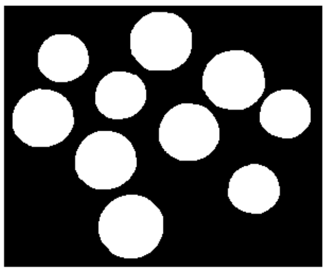


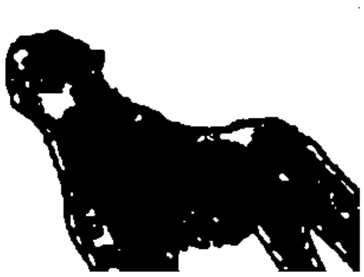


As illustrated above, we can clearly recognize the objects.

Now that we have the segmented color images, we can measure the similarity coefficient. To do so, we converted these images to binary. Here are the binary forms:





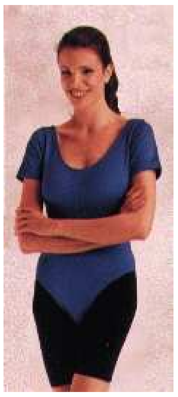


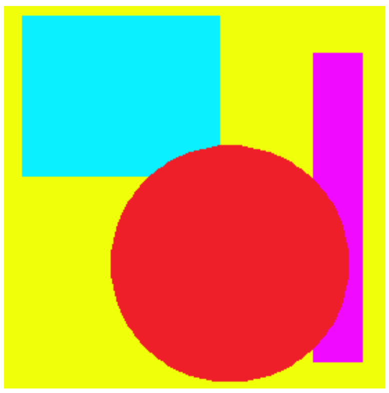
Now our images are ready to be compared with the ground truth. We'll explain that part in Results Analysis.

4.2) Color Images

In grayscale images, we considered only one intensity value, in other words gray level. But in color images, we have RGB components; hence we can consider the value of one pixel as a vector of 3 components and perform the calculations in this manner. This is what we did in our experiments. That is the only difference between the grayscale part.

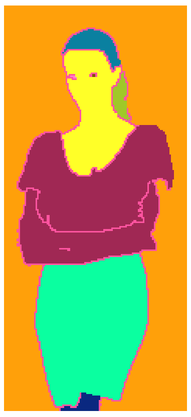
Here are the original color images before applying the algorithm:

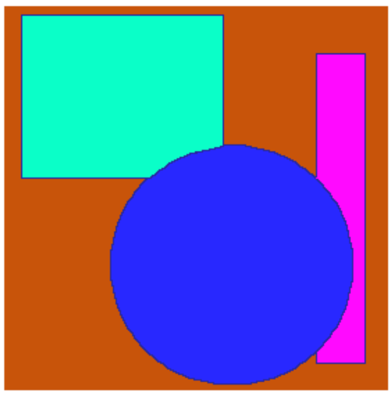

woman.tif


color.tif

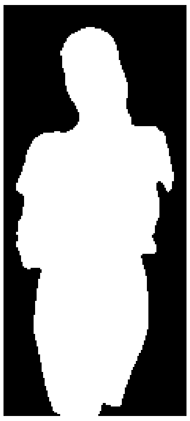
Note that there is an *exception* for color.tif. We have not applied randomized seeding to this image because it takes too much time. Instead, we manually placed the seeds in 4 different shapes.

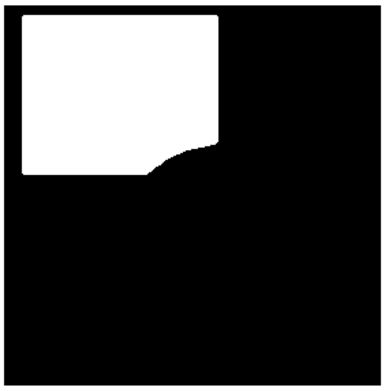
Here are the segmented images after applying the algorithm and color coding:





And finally, the results after converting to binary image:





As we can realize, color.tif gives us a poor binary result. That's because the colors are exact colors and the background cannot be distinguished from the shapes.

4.3) Results Analysis

For all the images we compared the ground truth and the obtained image. These comparisons gave us the quantities of True Positive (TP), False Positive (FP) and False Negative(FN). Hence, we applied the formula below and found the Dice Similarity Coefficient.

$$DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

Here is a table that shows the best DSC's obtained from our experiments:

	gris21.tif	coins.png	feli.tif	woman.tif	color.tif
DSC	0.9324	0.9512	0.9428	0.9717	0.5228

All of the image gave satisfactory results except color.tif. Since they are above 90%, that means the images are well-segmented.

5) Organization and Development of the Coursework

In development of this coursework, Region Growing were implemented and tested in two parts: segmentation and quality control. Segmentation part included the algorithm for both grayscale and color images while quality control part included the color coding, binary conversion and DSC calculation. Report was written collaboratively in one document. The approximate times for tasks can be given as:

- Analysis: 4 hours
- Implementation: 2 days
- Testing: 2 days
- Documentation: 1 day

6) Conclusion

As conclusion, we can say that Region Growing is a fast algorithm and it is easy to implement. Applying to both grayscale and color images, we obtained nearly perfect results. The problems we encountered were execution times (too long in some images) and automatic-manual trade-off. Because in some images, randomized seeding didn't work well. That has been a very fun and beneficial coursework. Learning, analyzing, implementing and making experiments on this subject widened our points of view and made us more curious about this field.

References

http://en.wikipedia.org/wiki/Segmentation_(image_processing) Accessed on 22.02.11

http://www.mathworks.com/help/ Accessed on 22.02.11