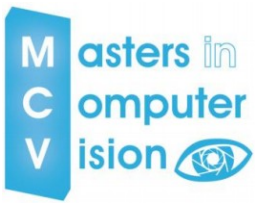


Advanced Image Processing

Lab 1

Scene Segmentation and Interpretation

Image Characterization

Alper Akcoltekin

Abdoulaye Diakité

31.01.2011

1) Introduction and Problem Statement

The main purpose of this lab is to analyse and implement different statistical methods in order to extract image texture descriptors. A texture can be defined as a region of an image that has similar characteristics. These characteristics are such a basic pattern that repeats, or frequency characteristics *[1]*. Then we can see that texture is a important image feature we can use in many computer vision system.

In this first part, we will try to clearly show the different textures of the set of pictures given to us, using the Gray Level Co-occurence Matrices (GLCM) and Energy Filters - which is also known as Laws' Masks. Our aim is to obtain texturized images using the statistical features of the images in both methods. Applying these methods give us convenience in image segmentation and object recognition which may be applied afterwards.

2) Algorithm Analysis

In this section, we analyze and describe the algorithms used for applying the texture descriptors to given images.

2.1) GLCM

This algorithm is quite simple with only one loop needed. After loading the picture to test, we define the specific transformation to the space we need (gray-level, RGB or HSV). A square matrix (called sub-image in our algorithms) of a chosen dimension is needed to define the number of gray-levels for the GLCM, then we just use predifined functions insede the loop to process pixel by pixel and to compute the results.

2.2) Energy Filters

In order to apply Energy Filters texture descriptor, we have two steps:

1) Convolve the image with one of Laws' masks

2) Obtain statistical measures from the results of convolution

Laws' masks are derived from primitive one dimensional filters given below:

L3 = (1 2 1), E3 = (-1 0 1), S3 = (-1 2 -1) and

L5 = (1 4 6 4 1), E5 = (-1 -2 0 2 1), S5 = (-1 0 2 0 -1), R5 = (1 -4 5 -4 1), W5 = (-1 2 0 -2 -1)

After choosing the appropriate mask, we convolve the image with it. Then we perform another convolution for computing the statistical values (**mean, absolute mean** and **standard deviation**) and obtaining the resulting image.

3) Design and Implementation

The algorithms were implemented and tested in MATLAB. The GLCM part is mainly based on two Matlab functions which are "**graycomatrix**", that create gray-level co-occurrence matrix from image and "**graycoprops**" to play around the contrast, energy and homogeneity properties of the co-occurrence matrices. The entropy is obtained using the functions "**entropyfilt**" and "**mat2gray**".

3.1) GLCM

Initialization of the picture
Taking the dimensions of the picture using two variables m and n
computing the entropy result

*for **i** going from the first to the last row of the picture*
    *for **j** going from the first to the last column of the picture*
        Definition of the sub-image
        *Computation of the GLCM using "**graycomatrix**" function and the sub-image*
        *Computation of the contrast, the energy, the homogeneity and the correlation using the results of the GLCM and the "**graycoprops**" function*
        *Creation of the output image pixel by pixel (using the current **i** and **j**)*
    end
end

3.2) Energy Filters

Given one of the Laws' masks and an image, we apply the algorithm given below:

Convolve the image I with Laws' mask;
FOR each pixel in convolved image I
    Extract the sub-image with a specific window size
    Compute the statistical value (mean, absolute mean or standard deviation)
    Set the pixel value to that statistical value
END FOR
Normalize the resulting image if necessary

4) Experimental Results and Results Analysis

4.1) GLCM

4.1.1) Grayscale

We mainly used a 3x3 sub-image to compute our results. And we played around three parameters of the "graycomatrix" Matlab function:

– the "***Offset***" that permit to specify the distance and the orientation between the pixel of interest and its neighbor. The following table shows the corresponding angle for a given distance D *[2]*.

Angle	**Offset**
0	[0 D]
45	[-D D]
90	[-D 0]
135	[-D -D]

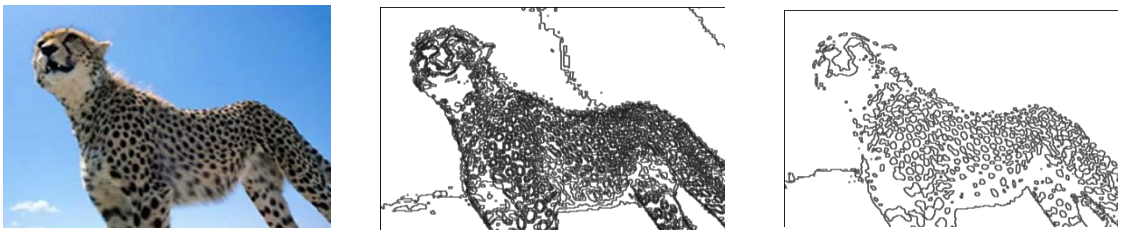
– The "***Numlevel***" that permit to specify the number of gray-levels to use when scaling the grayscale values in the sub-image. The values of this parameter go from 1 up to 8 *[2]*.

The third parameter is the "***Symmetric***" and it takes true or false value, computing the GLCM in both senses of the neighbor (for example from right to left and left to right in the case of an Angle 0 and Distance 1 GLCM). But after we saw that it doesn't give big differences between the output results, except it takes more time when it is in true value, we decided to keep it false (which is the default value by the way).

the **Entropy** results are computed using the Matlab function "***entropyfilt***" *[3]*, and are independent of the *"**graycoprops**"* function.

We tried first to see the outputs images given by the default configuration of the *"**graycoprops**"* function, using a **3x3** sub-image. And changing the 'Numlevel' parameter creates more separated regions, while the Entropy result doesn't change since it is independent of this function. As we can see in the following results, the different regions of the pictures are not enough uniform to be useful for a segmentation work.

**N = Numlevel*

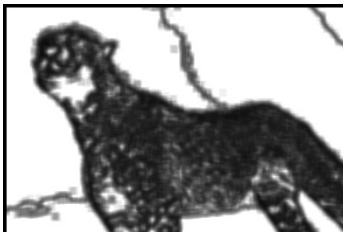


(a) (b) (c)

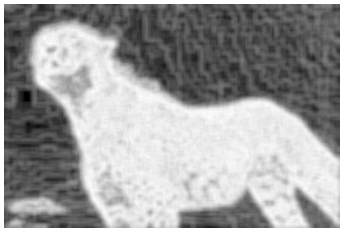
figure1: (a) is the original image (feli.tif), (b) and (c) are the Energy results respectively for *N* = '8' and *N* = '3'.

The entropy function gives the best results, and we can conclude that greater is the number of gray-levels used when scaling the grayscale, better are the results. Now if we considerably increase the window size of the sub-image (that correspond to the number of the neighbors taken in consideration) the results are globaly good.

9x9 Sub-Image

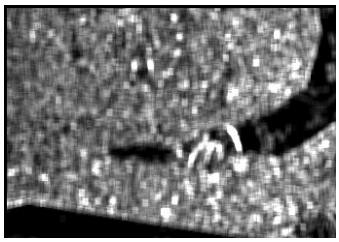


Energy result (*N* '8')

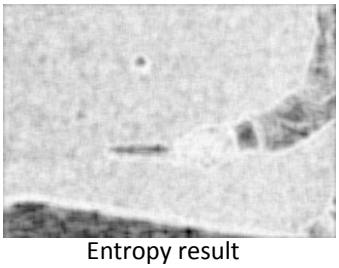


Entropy result

7x7 Sub-Image



Contrast result (*N* '8')

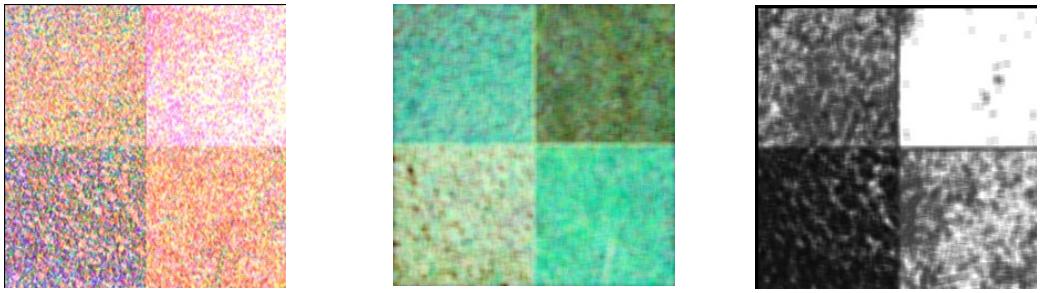


Entropy result

After testing some modification of the orientation angle and the distance using the ***Offset*** parameters, we found that the edges are more acquirated on the output picture when the value of the angle is increasing. But it is not good for the results since it creates white area on the edges and increase the number of regions for the segmentation.

4.1.2) Color spaces

We tested the algorithm on RGB and HSV color spaces. First we tried each channel of the RGB space, but they give almost same results as the grayscale, and summing them gives color images without changing the uniformity of the textures. So we focused on the HSV space, and this time, the Hue channel gives interesting results.

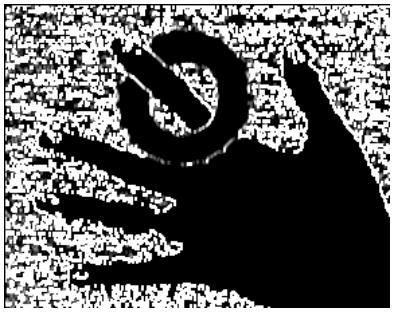


(a) (b) (c)

figure2: (a) Homogeneity result after summing the 3 channels of hsv color space, using angle 135, *N '5'* and 3x3 sub-image. (b) Entropy result using 7x7 sub-image.
(c) Energy result using 7x7 sub-image and the default configuration of *"**graycoprops**"*.

Once again, for the HSV color space, a 3x3 sub-image can give good results but the greater dimensions are better, and using the hue value we can get some results presenting uniform texture areas (almost already segmented pictures).

3x3 Sub-Image



Contrast result on hue (*N* '5')

Energy result on hue (*N* '5')

9x9 Sub-Image

Contrast result on hue (*N* '8')

Energy result on hue (*N* '8')

4.2) Energy Filters

We know that we have a variety of choices for mask selection; so this algorithm was tested with many masks and best results were selected. We also observed significant changes when we changed the window size for the second phase (obtainin statistical results) of the algorithm. Sometimes it was 7x7 and sometimes 10x10.

We also tested images in different color spaces such as RGB and HSV. Some images gave very satisfactory results in these color spaces because of their characteristics. We will present each experiment in seperate sections.

Our another observation was that among the three statistics – mean, absolute mean and standard deviation, the best results were obtained mostly from absolute mean.

4.2.1) Grayscale

Applying our algorithm to several images after converting to grayscale gave us good results in the images below in Fig. 1.

(a) (b) (c)

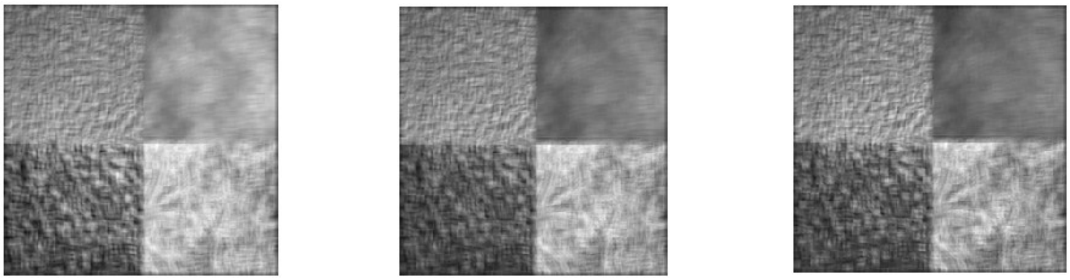
Figure 3: Energy Filters texture descriptor applied on feli.tif, hand2.tif and mosaic8.tif. (a) is the result for **mean statistics** while (b) and (c) are the results for **absolute mean statistics**.

Although we obtained good results from these images, it's not always sufficient to clearly see that the image is segmented into parts. In order to apply segmentation after texturizing the image, we need to have the results as segmented as possible. W5L5 mask was used in this test and 7x7 windows size were used for computing statistical values.

4.2.2) RGB Color Space

In this experiment, we extracted red, green and blue channels from the original image and applied this algorithm on each channel. While each channel sometimes gave different results, we also combined the channels again to see the output in a color image again.

The results for seperate channels were illustrated below in Fig. 2. As we can see, applying our algorithm on seperate channels of RGB gives us very similar results on mosaic8.tif.



(a) (b) (c)

Figure 4: Images represent each channel. (a) is red, (b) is blue and (c) is green.

The result after combining each channel and constructing a colored image is illustrated below in Fig. 3.

Figure 5: Colored result of the algorithm on mosaic8.tif

Here is another result with feli.tif which is obtained by blue channel of the image.

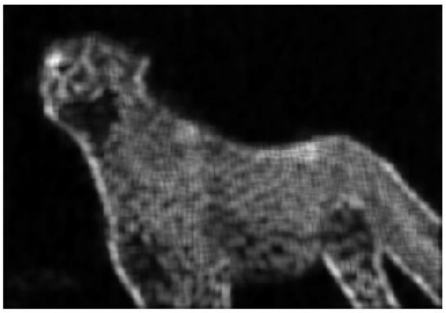
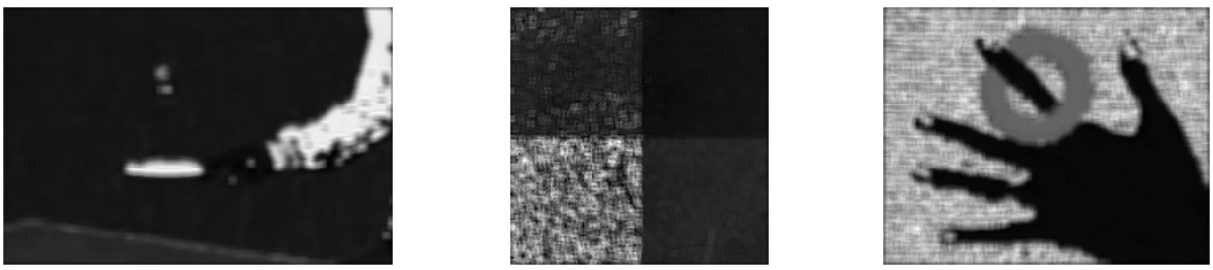


Figure 6: Blue channel of feli.tif

Actually, resulting color image looks very similar to the original image; but anyways it's not the best result. This test was performed on **mosaic8.tif** with the W5W5 mask and provided results are the ones obtained with **mean statistics**. It was performed on **feli.tif** with the W5E5 mask and provided results are the ones obtained with **absolute mean statistics**. 7x7 window size was used for statistics computation.

4.2.3) HSV Color Space

HSV is color space where components are less correlated to each other; so each component of an HSV image (Hue, Saturation and Value) represents a different feature of the image. We know that Hue value represents color; thus it can easily be predicted that we can obtain some concrete results if we convert our image to HSV space and apply our algorithm to the Hue value of the original image. This test gave very satisfactory results on several images. Fig. 6 illustrates these results.



(a) (b) (c)

Figure 7: Results for Hue values for different images.

As we can see in Fig.6, segments inside the images are easily distinguishable. In (a), we can easily recognize the tennis racquet, the ball, the jacket of the player and the tennis table. In (b), each area is in different colors. In (c), background, the hand, the ring and even the nails are easily recognized. This test was performed with W5W5 mask using 7x7 window size.. Hence, we can conclude that we got the best results from HSV color space and that these results can be very useful for further processing i.e segmentation and object recognition.

4.3) Comparison and Analysis

For the GLCM algorithm, the best configuration is probably the 0 angle and "***Numlevel***" '8', and choosing a sub-image with a dimension going from 7x7 to greater gives good results. Most of the best results are given by the entropy parameter, and the computation time is relatively long, since the algorithm use a loop and have to compute 4 differents parameters for each pixel.

Entropy: very good

Energy: good

Contrast: average

Homogeneity: bad

Computation time: 6 to 15 min

For Energy Filters texture descriptor, the tables below indicate the comparison of each statistics in the means of quality and speed.

	Quality	**Approximate Time (ms)**
Mean	Good	0.152872
Absolute Mean	Very Good	0.153059
Standard Deviation	Poor	0.225718

It is obvious that Laws' Masks method is pretty fast in all of the statistics. Absolute mean computation generally gave us the most satisfactory results. Furthermore, we can say that the best results were obtained using **HSV** color space.

5) Organization and Development of the Coursework

In development of this coursework, two methods were shared, implemented and tested seperately. Later, the results were combined together and discussed. Report was also written in the same manner and it was later combined in one document. The approximate times for tasks can be given as:

- Analysis: 2 hours
- Implementation: 4 hours
- Testing: 3 days
- Documentation: 2 days

6) Conclusion

For the GLCM algorithm, we can conclude that greater is the size of the sub-image, better are the results since they will be more uniform regions. On the other hand, we had better keep the angle at 0, or closer if we don't want to create extra regions on the edges. Anyway, it can be interesting, for some other process different from the segmentation, to study the low gray scaled outputs (Numlevel at 3 for example) because the results shows real accurated texture, but not uniform at all. Meaning the it can be useful to detect edges inside a texture.

For Energy Filters (Laws' Masks) method, many different masks were tested to find the best result. As a result, different masks gave different results on each image; hence, best results were obtained with different masks on each image. Considering the three statistics that were applied on the images, absolute mean was the one that gave very clear results. We can also conclude that after testing with different color spaces, HSV gave the most satisfactory results in most of the images.

Globaly, we enjoyed this interesting lab, and we learned more about the texture of a picture. And performing some research, trying to find and understand the solutions of the problems train us to be good researchers in the near future.

References

[1] *http://fr.wikipedia.org/wiki/Texture*

[2] *Matlab help*

[3] *http://www.mathworks.com/*