Image Characterization using Texture
LAB 02

Taner GUNGOR
Al ARAFAT

March 25, 2015

Deadline: March 04, 2015
Instructor: Xavier Llad
Arnau Oliver

1 Introduction and problem definition

1.1 Introduction

Image characterization involves in collecting the image properties that represent regions of the image based on different criteria. These criteria can depend on several particularities of surface and structure of the image. Texture analysis is one of them. In computer vision, texture analysis plays a vital role because it constitutes a major step in segmentation, classification and recognition. Thee textures can be different and since they possess very different characteristics, there is no generic texture model that can properly describe them. Several methods have been proposed and used to characterize them but none of are purely perfect.

The Statistical methods for characterizing texture, analyze spatial distribution of gray values by computing local features at each point in the image and deriving statistics from the distributions of the local features. In this lab experiment, the most used statistical methods as co-occurrence matrices and energy masks have been used to characterize texture. Computing co-occurrence matrices is fast, easy and also provide good results that can be used for image segmentation and object recognition.

The co-occurrence matrices measure the probability of finding two pixels with certain values in a direction inside a window. Then, the statistics like energy, entropy, contrast, homogeneity, probability, correlation are computed. At first, masks have been convoluted over the image and then the statistic measures are obtained from the convolution result. Energy masks filter small windows with predefined masks and then the mean and standard deviations are computed. Finally texture descriptors using Grey Level Co-occurrence Matrix (GLCM) and the law filters (masks) have been generated. Texture can be computed globally or locally. In global computation, all pixels of an image are used to get the vector of features. While in local computation, for each pixel, using the neighbors of the pixel, the feature vectors are computed. Local computation involves in segmentation while global computation involves in classification.

These descriptors are to be used for segmentation which are different for different textures. Texture descriptors created by GLCM are the statistics of the co-occurrence matrix for each pixel neighborhood in the image.

1.2 Problem definition

In this lab experiment, the main objective is to enhance the performance of the region growing algorithm by computing textures of the images and generating output images to segment them. The images to test for this lab experiment have been provided with the lab. Co-occurrence matrices have been generated and some statistics have been computed to apply at local level to produce descriptors for each pixel. These descriptors are the feature vectors for each pixel. Finally, the feature vectors have been used to classify the dataset textures in different classes. This enhance the performance of the region growing segmentation algorithm's performance. The steps that define the problem are,

- Compute co-occurrence matrix.
- Compute some statistics (contrast, energy, homogeneity and entropy) from the matrices and create corresponding output images.
- Add some statistics with the RGB images from the region growing algorithm to get better result.

- Compare the resultant images with the images, we got from the region growing algorithm.
- Generate the vector of features for an image and use this information to classify the images in different classes. The script *scr_classify* that uses the weka library has been provided to classify the images of the dataset with one classifier called Random Forests.

Then the co-occurrence matrices for the images have been generated globally to classify the dataset.

2 Algorithm analysis

Steps that have been followed to perform the image characterization starts from computing the co-occurrence matrices. Then different statistics have been computed from the matrices. Finally the vector of features are generated to classify the images. A brief overview of the whole algorithm is described below.

2.1 Co-occurrence matrices

A co-occurrence matrix is a matrix that is defined over an image to find the distribution of co-occurring values at a given offset.[1] That means, the co-occurrence matrix represents the probability of finding pixels with the same intensity value at a certain oriented distance. Co-occurrence matrices are based on second-order statistics i.e. on the spatial relationship between pairs of pixels with certain gray values in the digital texture images. The co-occurrence matrix is usually referred to GLCM (Gray Level Co-occurrence Matrix) which contains the information regarding different frequency combination of pixels occurring in an image. Mathematically, a co-occurrence matrix A is defined over an n × m image I, parameterized by an offset (Δx,Δy), as:

$$A(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{cases} 1 & \text{if} I(p,q) = i \text{and} I(p+\Delta x,\ q+\Delta y) = j \\ 0 & \text{otherwise} \end{cases}$$

here i and j are the intensity values of the image, p and q are the spatial positions in the image I and the offset (Δx, Δy) depends on the direction, θ and the distance at which the matrix is computed, d. Co-occurrence matrices are sensitive to rotation. Thus the co-occurrence matrices are formed using a set of offsets sweeping across 180 degrees (i.e. 0, 45, 90, and 135 degrees) at the same distance to achieve a degree of rotational invariance [2.].

2.2 GLCM

When the co-occurrence matrices are computed based on gray level values are called Gray Level Co-occurrence Matrices. Gray Level Co-occurrence Matrix is a tabulation of how often different combinations of pixel brightness values (gray levels) occur in an image. GLCM can be computed globally or locally. For the local calculation, a window of size n × n has been defined. The size of the window depends on the texture and should be big enough to get the texture properties. For the central pixel of that window, the GLCM matrices are calculated. The next step includes shifting the window one pixel and computing the GLCM for the central pixel. This process continues until computing GLCM for each all the pixels in the image.

To get the GLCM matrices, several parameters are considered. They are,

- The window size of the neighborhood for a pixel. If the window size is selected small where the texture of the image contains small regions then the final segmentation result might be over segmented. The image will be under segmented if the selected window size is larger than the texture size of the image.
- For each window there are values for orientation, θ and distance, *d*.
- Then number of gray levels.

The orientation 0°, 45°, 90° and 135° can be used to construct the GLCM matrix. The angels for 3 × 3 neighborhood can be shown as,
The above figure illustrates the array: offset = [0 1; -1 1; -1 0; -1 -1]. In MATLAB the representation is different. MATLAB represents offset as,

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

in offset, the 1^{st} part represents the orientation and the 2^{nd} part represents the distance.

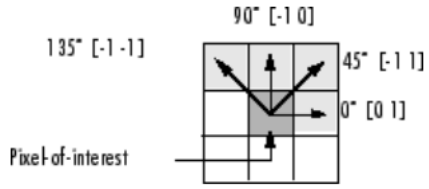

Figure 1: Orientation for the GLCM matrix. [3]

These angels represent the bidirectional orientation, as example, finding the neighborhood for 0° is equivalent to finding the neighbor in the 180° direction. The distance is given in numbers of pixels. The distance is same in all direction for all the neighborhood pixels from the middle pixel.

To construct the co-occurrence matrix, the number of times that a pixel pairs appear within the neighborhood for specified orientation and distance, is counted for each pixel. A position in the matrix (i,j) corresponds to the number of pixels of intensity i and j separated by a given distance and angle. They are always square matrices with the same dimensionality as the number of gray-levels. An example of computing the co-occurrence matrix is shown below.

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Figure 2: Image matrix.[1]

For this image matrix, the GLCM with distance $d = 1$ and $\theta = 90°$ can be computed as

- In the neighborhood the number of pixels pairs at a certain orientation and certain distance is computed.
- For 0,0 pair at 90° orientation, there are 6 pairs. So, the value for the 1_{st} pixel will be 6 for distance 1.
- This process continues for all the pixels in the neighborhood.

Thus the final GLCM matrix, M_v for this matrix will look like,

$$M(1, 90°) = \begin{bmatrix} 6 & 0 & 2 & 0 \\ 0 & 4 & 2 & 0 \\ 2 & 2 & 2 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix}$$

After computing the co-occurrence matrix the texture descriptors are described using statistics.

2.3 Texture Descriptor

The statistics used in this algorithm are the contrast, energy, homogeneity, and entropy. They are described below.

Contrast: Contrast indicates the local variation in the image intensity between a pixel and it's neighbor.

$$\text{big value} = \text{large variation.}$$
$$\text{small value} = \text{uniform}$$

The equation to compute the contrast is:

$$contrast = \sum_{i,j=1}^{(N-1)^2} m_{i,j} \times (i-j)^2$$

Energy: Energy indicates the uniformity of the image.

big value = few entries in the matrix with large values.
small value = non-uniform image *i.e.* entries in the matrix are with similar values.

The equation to compute the energy is:

$$energy = \sum_{i,j=1}^{(N-1)^2} m_{i,j}^2$$

Homogeneity: Homogeneity measures the values concentrated in the diagonal of the matrix *i.e.* spatial closeness of the distribution of the co-occurrence matrix.

big value = homogeneity.
small value = uniform.

The equation to compute the energy is:

$$homogeneity = \sum_{i,j=1}^{(N-1)^2} \frac{m_{i,j}}{1+|i-j|}$$

Entropy: Entropy measures the randomness of the elements of the co-occurrence matrix.

big value = elements in the matrix are equal.
small value = all elements are different

The equation to compute the entropy is:

$$entropy = \sum_{i,j=1}^{(N-1)^2} m_{i,j} \times log_2(m_{ij})$$

All these statistical features can be calculated both locally and globally. In case of global calculation, the matrix for the whole image are calculated at once, rather than using windows. Locally extracted features are used for image segmentation and global extracted features are used for classification.

3 Design and implementation

The algorithm has been implemented in steps. MATLAB function *graycomatrix* has been used to compute the co-occurrence matrix using parameters predefined by the user and the *graycoprops* function has been used to compute all the statistics describing the texture of the neighborhood.

- Choosing the size of the window. For this experiment, 3 different sizes of windows have been used. 10 × 10, 15 × 15 and 20 × 20.
- For each window, the co-occurrence matrices and different statistics have been computed.
- Finally the image has been normalized. The normalization is performed by dividing all the elements of the matrix by the maximum value available in that matrix. The steps are shown below.

4 Experimental section and results

After running the algorithm, we computed the GLCM and then the statics for the matrices. The criteria upon which we changed our input images are

- orientation,
- distance,
- window size.

The output images for different measurement are given below.

For **Image 1, feli**,

- Window size: 10 × 10.
- Orientation: 45°.
- Distance: 1.
- Statistic: Energy.

Figure 3: energy statistic for feli with [-1 ,1]

- Window size: 10 × 10.
- Orientation: 45°.
- Distance: 1.
- Statistic: Homogeneity.

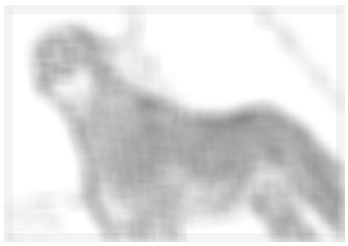

Figure 4: homogeneity statistic for feli with [-1 ,1]

- Window size: 10 × 10.
- Orientation: 0°.
- Distance: 3.
- Statistic: Energy.

Figure 5: energy statistic for feli with [0 ,3]

- Window size: 10 × 10.
- Orientation: 0°.
- Distance: 3.
- Statistic: Homogeneity.

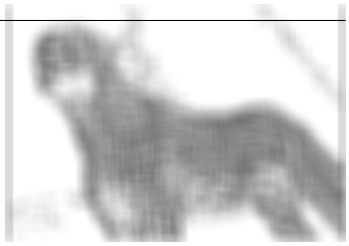

Figure 6: homogeneity statistic for feli with [0 ,3]

- Window size: 15 × 15.
- Orientation: 0°.
- Distance: 3.
- Statistic: Energy.

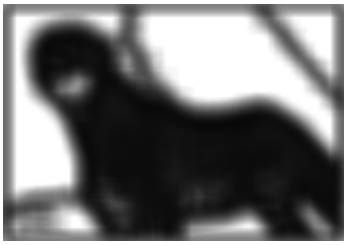

Figure 7: energy statistic for feli with [0 ,3]

- Window size: 15 × 15.
- Orientation: 0°.
- Distance: 3.
- Statistic: Homogeneity.

Figure 8: homogeneity statistic for feli with [0 ,3]

- Window size: 20 × 20.
- Orientation: 0°.
- Distance: 3.
- Statistic: Energy.

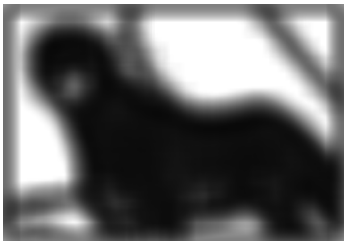

Figure 9: energy statistic for feli with [0 ,3]

- Window size: 20 × 20.
- Orientation: 0°.
- Distance: 3.
- Statistic: Homogeneity.

Figure 10: homogeneity statistic for feli with [0 ,3]

For **Image 2, hand**,

- Window size: 10 × 10.
- Orientation: 90°.
- Distance: 1.
- Statistic: Contrast.

Figure 11: contrast statistic for hand with [-1 ,0]

- Window size: 15 × 15.
- Orientation: 90°.
- Distance: 1.
- Statistic: Contrast.

Figure 12: contrast statistic for hand with [-1 ,0]

- Window size: 15 × 15.
- Orientation: 90°.
- Distance: 3.
- Statistic: Correlation.

Figure 13: correlation statistic for hand with [-3 ,0]

- Window size: 20 × 20.
- Orientation: 90°.
- Distance: 1.
- Statistic: Contrast.

Figure 14: contrast statistic for hand with [-1 ,0]

Segmentation Result from texture characterization
To perform texture characterization using statistics, the region growing segmentation algorithm developed in the 1^{st} lab has been used. The input images to the region growing algorithm are the RGB image planes along with Entropy, Energy, Correlation, Homogeneity and Contrast descriptors (everything normalized). Threshold value is also provided to get the best possible output.

We implemented our algorithm over the *feli* image. Our experiment output is shown below.

- Window size: 10×10.
- Orientation: $0°$.
- Distance: 1.
- Statistic used : Energy and Homogeneity.
- Threshold: 40
- 1^{st} image is the original image, 2^{nd} image is the output from the region growing algorithm and 3^{rd} image is the output by using statistics with RGB layer *i.e.* RGB + E + H.

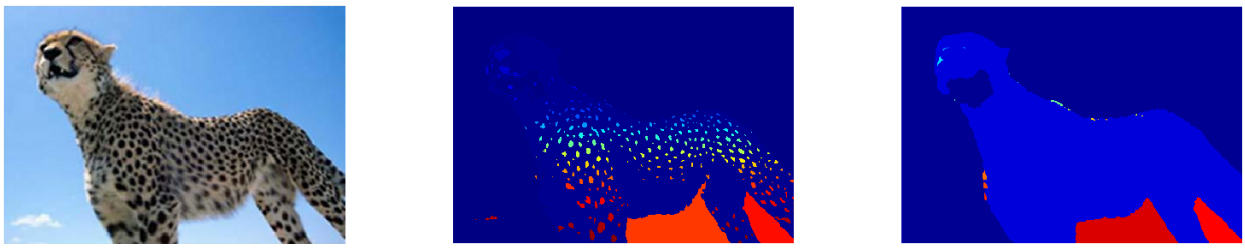

Figure 15: Segmentation using texture characterization

- Window size: 20×20.
- Orientation: $0°$.
- Distance: 1.
- Statistic used : Energy and Homogeneity.
- Threshold: 40
- 1^{st} image is the original image, 2^{nd} image is the output from the region growing algorithm and 3^{rd} image is the output by using statistics with RGB layer *i.e.* RGB + E + H.

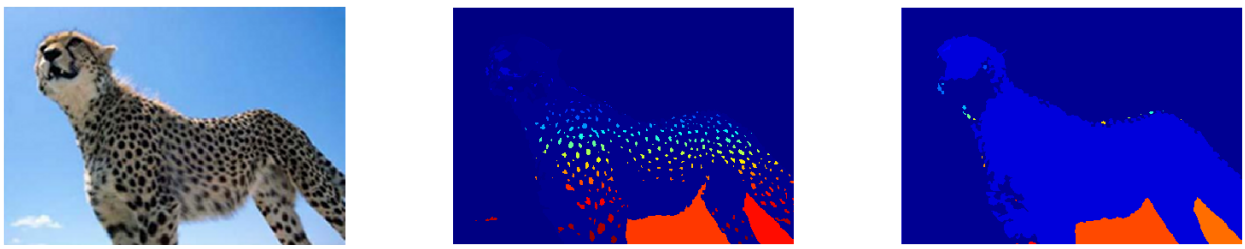

Figure 16: Segmentation using texture characterization

Then we applied median filter over the original image to improve the result. The results are given below.

- Window size: 10 × 10.
- Orientation: 45°.
- Distance: 1.
- Statistic used : Energy and Homogeneity.
- Threshold: 40
- 1^{st} image is the original image, 2^{nd} image is the output from the region growing algorithm and 3^{rd} image is the output by using statistics with RGB layer *i.e.* RGB + E + H.

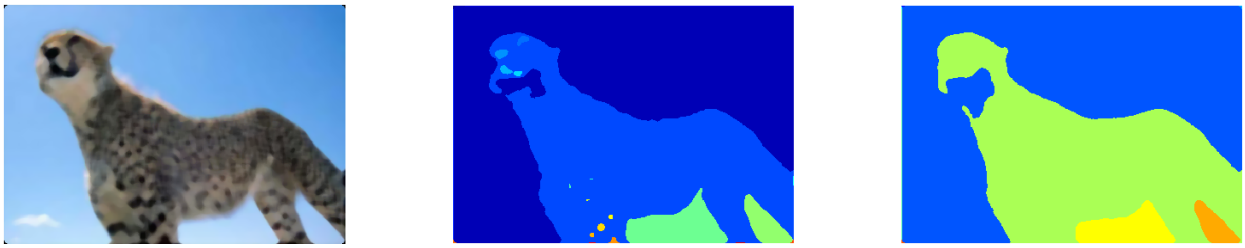

Figure 17: Segmentation using texture characterization over filtered image

---

- Window size: 15 × 15.
- Orientation: 45°.
- Distance: 1.
- Statistic used : Energy and Homogeneity.
- Threshold: 40
- 1^{st} image is the original image, 2^{nd} image is the output from the region growing algorithm and 3^{rd} image is the output by using statistics with RGB layer *i.e.* RGB + E + H.

Figure 18: Segmentation using texture characterization over filtered image

- Window size: 20 × 20.
- Orientation: 45°.
- Distance: 1.
- Statistic used : Energy and Homogeneity.
- Threshold: 40
- 1^{st} image is the original image, 2^{nd} image is the output from the region growing algorithm and 3^{rd} image is the output by using statistics with RGB layer *i.e.* RGB + E + H.

Figure 19: Segmentation using texture characterization over filtered image

The comparison between region growing algorithm and segmentation using texture characterization algorithm Comparison Results :

- **Region growing :** The region growing algorithm is fast but does not provide more freedom in segmentation. It also does not manipulate good result in segmentation.

- **Image Segmentation using texture characterization :** Image segmentation using texture characterization provides more degree pf freedom in selecting parameters. Thus it provides better result in segmentation. But this algorithm is time consuming. It requires a huge amount of images to operate as for different statistics it generates different images. Despite of time complexity, the algorithm performs well in segmentation.

Dataset classification :

Finally, we had to classify the dataset textures. The classifier *scr_classif y.m* was provided to classify the images of the *p2_class* folder. The script uses the *weka* library to classify the images of the dataset with one classifier called *RandomForests*. *Weka* is a collection of machine learning algorithms for data mining tasks. The algorithm can either be applied directly to a dataset or called from users own Java code.

We also used the *scr_classifyPR.m* to classify using The *PRTools* supplies more than 300 user routines for traditional statistical pattern recognition tasks. It includes procedures for data generation, training classifiers, combining classifiers, features selection, linear and non-linear feature extraction, density estimation, cluster analysis, evaluation and visualization. First we modified the function *computeFeatureVector* to produce a feature vector that has higher accuracy.

The matrix we got is as following,

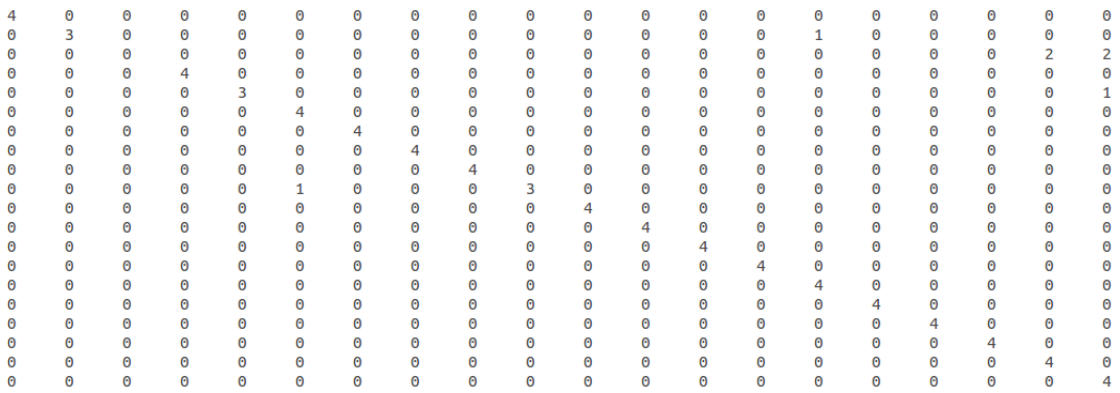

Figure 20: Matrix form dataset classification using texture characterization.

The answer for the dataset classification is : 0.9125
And the time is 13.055558 seconds.
The maximum accuracy we got using Weka library *scr_classif y.m* is 76.75%. On the other hand, using *PRtools scr_classifyPR.m* we got 91.25%. We got the better accuracy rate for classification using *PRtools* much higher than the Weka classifier.

5 A brief discussion regarding the algorithms

While we were working on the region growing algorithm we noticed some advantages of the this algorithm and also some of it's disadvantages. A brief pointing to the advantages and disadvantages of region growing is given below.

Advantages

1. It makes the region growing algorithm more efficient.
2. If the initial parameters can be chosen perfectly then it would provide better result.
3. It provides better result in dataset classification.
4. Various mathematical operations can be used to alter, compare and transform textures. Most available texture analysis algorithms involve extracting texture features and deriving an image coding scheme for presenting selected features.

Disadvantages

1. The algorithm depends on the size of the window.
2. This algorithm is not time efficient.
3. Choosing the correct statistical method plays a vital role.

We used median filter over some images to improve the output result.

6 Organization and development

In this section, we have presented the management process we followed to organize and develop a deliverable as required by the lab work. To accomplish the assigned task, we made a plan to organize our lab work. We arranged and synchronized our management process considering several criteria such as

1. Studying and analyzing the problem,
2. Defining the approach by relating it with the previous lab experiment,
3. Coding the segmentation and classification using texture analysis,

4. Testing the code,

5. Report writing.

As this lab relates with the 1^{st} experiment, we stated with finding the relation of this lab experiment with the previous one and finding the point from where we had to start. We analyzed the problem before the lab and we computed the co-occurrence matrix with statistics in our first lab. Before the second lab, we had implemented the algorithm over RGB color images to get the segmented images. on the second lab day, we compared our result with the results from the first lab and finished the classification portion.

7 Conclusions

After finishing the lab experiment we can conclude that the algorithm we used is pragmatic. The texture characterization of images using co-occurrence matrix can be carried out using different statistics. Choosing different statistic changes the result as it offers several degree of freedom. Thus, the acceptable result can be achieved by examining different parameters. Although the experiment is time consuming, it was very interesting and challenging.

References

[1] Class lecture

[2] **Steven W. Zucker, Demetri Terzopoukos**, *Finding Structure in Co-Occcurrence Matrices for Texture Analysis*, Computer Vision and Image Processing 12, 286-308(1980).

[3] $http://es.mathworks.com/help/images/ref/graycomatrix.html$

[4] $http://en.wikipedia.org/wiki/Co-occurrence_matrix$