Master Computer Vision

Scene Segmentation and Interpretation

Image Characterization

Author:
Muhammad Usman
Emre Ozan Alkan
Priyanka Phutane

Course Co ordinator:
Pere Ridao Smith

1 Introduction and Problem Statement:

In everyday life Texture means roughness, smoothness etc. which refers to feel. A texture which is considered as rough has a large difference between its highs and lows. This difference will be close to size of fingers so one feels it. Similarly for smooth texture this difference will be very small.It works similarly in Images the difference is now highs and lows are brightness values.So In imaging world texture means a certain pattern which occurs repetitively in images.[1] Texture analysis can be used for image segmentation and can be used in the variety of applications, including remote sensing, automated inspection, and medical image processing.[2] The goal of this lab is to study and implement the statistical methods to extract the image texture descriptors which can be used further for image segmentation.
In this lab we will use two methods for the extracting the image descriptors. In first method we will use the Grey Level Co-occurrence Matrix (GLCM)[2] and extracts values for Contrast, Homogeneity, Energy, and Entropy. As our goal will be to display these texture descriptors in the form of image so we will calculate them for every pixel and display the corresponding images. In the second part we are using the Energy Filter (Law mask) which gives us the static measures (mean, abs mean, and standard deviation) which are derived for each pixel and corresponding results are displayed.

2 Algorithm Analysis:

Now we will discuss the two algorithms we used for calculating texture descriptors.

2.1 Gray Level Co-occurrence Matrix

The GLCM is a tabulation of how often different combinations of pixel brightness values (grey levels) appear in an image.[1] First order texture measures are calculated from the original image values e.g. variance and do not depend upon the relation of pixel with its neighbourhood. Second order texture measures depend upon the relationship between group of two pixels. The Algorithm described here will be used to calculate second order texture measures.[1]

2.1.1 Spatial Relationship between two pixels:

GLCM algorithm considers the relationship between two pixels at a time. One pixel is called reference and other is called neighbour. One can choose neighbourhood by mean of vector and then each pixel in window (which is specified and changes will be considered inside that window only) will become reference and by doing so entries of GLCM matrix will be filled.[1]

2.1.2 Separation between two pixels:

One can also specify the distance between reference and neighbourhood pixels to be considered while calculating GLCM commonly known as offset. Commonly we use 1 as offset which means immediate neighbour of reference pixel. Care should be taken while selecting offset as it should be less than window size.[1]

2.1.3 Other Parameters:

The Co-occurrence matrix also depend upon window size we will use to compute GLCM matrix for certain pixel. It also depend upon the number of grey levels we want to use to build matrix as there can

be wide range of grey levels present in image and if we dont reduce the size of image then the size of co-occurrence matrix will be big so we want to scale down to make computation faster.[1]

2.1.4 Computing GLCM Matrix:

We construct our Co-occurrence matrix by simply considering our window specified by window size parameter. We take its centre pixel and move it to pixel for which we want to measure co-occurrence matrix. Now we will count pairs of pixels specified by taking into account the specified offset and spatial relationship.[1]

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

Table 1: Sample Image

Now we will show the co-occurrence matrix using distance 1 and Angle 0. One thing should be noted that we have taken care of symmetry while building this matrix. It will be different if we don't consider symmetry.

Neighbourhood Pixel → Reference Pixel ↓	0	1	2	3
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

Table 2: Co-occurrence Matrix using Distance 1 and Angle 0 deg [1]

2.1.5 Reading GLCM Matrix:

The first(top left) element of the matrix will always be filled with number of times combination (0,0) occurs which means how many times within image area a pixel with grey level 0(neighbour pixel) appears to the right of another pixel with grey level 0(reference pixel).[1] Different co-occurrence matrix exist for each spatial relationship.

Now we have computed our co-occurrence matrix for pixel on which our window was centred. Now we can use this matrix to measure different statistics given below.
Contrast:
Contrast is the measurement of the difference between pixel value and its neighbour value computed for entire image.Higher values of contrast indicate large variation in image intensities while low value indicate image is quite homogenius having small changes in intensities. Formula for calculating contrast is given below.

$$Contrast = \sum_{i,j=0}^{N-1} (i-j)^2 \times m_{ij}$$

Homogeneity:
Homogeneity is the quantitative measure of state of co-occurrence matrix being homogenius or uniform.

The formula for calculation of homogeneity is given below.

$$Homogeneity = \sum_{i,j=0}^{N-1} \frac{m_{ij}}{1 + |i - j|}$$

Entropy:
Entropy is the measurement of randomness among the co-occurrence matrix elements. The formula for calculation of entropy is given below.

$$Entropy = \sum_{i,j=0}^{N-1} m_{ij} \times \log\left(m_{ij}\right)$$

Energy:
The energy of co-occurrence matrix is calculated by the formula given below.

$$Energy = \sum_{i,j=0}^{N-1} m_{ij}^2$$

2.2 Energy Filters:

We can measure the texture energy by using Laws' method of texture energy transforms. To do so first of all we convolve our image with 2d Law's Mask. The output form these texture energy filters. These consist of moving window calculation of variance or more cheaply a moving window average of absolute values.[3] The 2d kernels used for convolution are primarily derived from one dimensional filers given below.

1 x 3 Filters	1 x 5 Filetrs
L3 = [1 2 1]	L5 = [1 4 6 4 1]
E3 = [-1 0 1]	E5 = [-1 -2 0 2 1]
S3 = [-1 2 -1]	S5 = [-1 0 2 0 -1]
	R5 = [1 -4 5 -4 1]
	W5 = [-1 2 0 -2 -1]

Table 3: Primitive 1 Dimensional Filters

3 Design and Implementation:

We implemented both algorithms in Matlab. Now we will discuss the implementation issues of both algorithm individually.

3.1 GLCM Algorithm:

As we stated before we used Matlab to implement our algorithm so we take benefit of built in Matlab function to implement our GLCM Algorithm. We used two important Matlab functions named "graycomatrix", which returns co-occurrence matrix for image according the parameters passed to it and "graycoprops" function which provide us required statistical measures for the passed co-occurrence matrix. Now we will explain the different parameters of these functions.

3.1.1 Parameters of "graycomatrix"

Below is the explanation of different input parameters of "graycomatrix".
Offset:
It is used to specify the distance and the orientation from pixel of interest(reference pixel) and the neighbour. The following table shows the orientation and corresponding offset for a given distance.

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

Table 4: Offset and Orientation Chart

Numlevel:
This parameter allows us to set the number of grey levels to be used while scaling the grey level values to sub image.It can take values between 1 and 8.[2]
Symmetry:
This parameter take two values true or false. If it is set then Matlab take into account the ordering of values in pixel pairs.[2]

3.1.2 Parameters of " graycoprops"

This function take GLCM matrix as input and name of desired property we want to find. The properties are Contrast, Energy, Correlation and Homogeneity.
Entropy of the Image can be calculated using Matlab function "**entropyfilt**" and is independent from graycoprops" function.
The Pseudo Code for our Implementation of GLCM is given below.

Data: Gray scale 'image'.
Result: Entropy, Contrast, Correlation, Energy and Homogeneity images of the input 'image' .
Calculate 'entropyImage' with 'entropyfilt' function; Initialize 'windowSize' with odd number preferably 7 or 9;
Initialize 'windowCenter' with floor of half of 'windowSize';
Pad the 'image' borders with size of 'windowCenter';
Initialize 'offset' with vector in form of [0 D], [-D, D], [-D, 0,], [-D, -D];
Initialize 'numLevels' with 8;
Initialize 'symmetric' with false or true;
Initialize 'contrastImage' as size of 'image' with zeros;
Initialize 'correlationImage' as size of 'image' with zeros;
Initialize 'energyImage' as size of 'image' with zeros;
Initialize 'homogeneityImage' as size of 'image' with zeros;
Initialize 'rows' and 'cols' with 'image' size;
for $i \leftarrow (`windowCenter' + 1)$ **to** $(`rows' - `windowCenter')$ **do**
  for $j \leftarrow (`windowCenter' + 1)$ **to** $(`cols' - `windowCenter')$ **do**
    Get 'subImage' as size of 'windowCenter' from 'image' with center i and j;
    Compute 'GLCM' with 'graycomatrix' function as input 'subImage';
    Compute 'contrast', 'correlation', 'energy' and 'homogeneity' with 'graycoprops' function as input 'GLCM' Set 'contrastImage' pixel(i, j) with 'contrast';
    Set 'correlationImage' pixel(i, j) with 'correlation'; Set 'energyImage' pixel(i, j) with 'energy'; Set 'homogeneityImage' pixel(i, j) with 'homogeneity';
  end
end
Remove paddings of 'image', 'contrastImage', 'correlationImage', 'energyImage' and 'homogeneityImage';

Algorithm 1: GLCM Algorithm

3.2 Energy Filters:

The pseudo Algorithm is given below.

Data: Colored 'image'.
Result: Laws' filter ans statistics of the input 'image' .
Separate RGB channels of the 'image' as 'imageR', 'imageG', 'imageB';
Initialize 'imageGray' with gray level of the input 'image';
Initialize 'hsvImage' with color conversion from RGB of 'image';
Initialize 'hueImage' from 'hsvImage"s first channel;
Initialize lawsMask with some mask;
Convolve 'imageGray', 'imageR', 'imageG', 'imageB', and 'hueImage' with Laws' mask;
Show the results;
Initialize 'filterSize' with odd number preferable 7 or 9;
Find mean of the 'imageGray', 'imageR', 'imageG', 'imageB', and 'hueImage';
Initialize 'meanOutputImage' with combination of mean of 'imageGray', 'imageR', 'imageG';
Show the mean results;
Find absolute mean of the 'imageGray', 'imageR', 'imageG', 'imageB', and 'hueImage';
Initialize 'absmeanOutputImage' with combination of absolute mean of 'imageGray', 'imageR', 'imageG';
Show the absolute mean results;
Find standart deviation of the 'imageGray', 'imageR', 'imageG', 'imageB', and 'hueImage';
Initialize 'stdDevOutputImage' with combination of standart deviation of 'imageGray', 'imageR', 'imageG';
Show the standart deviation esults;

Algorithm 2: Energy Filters Algorithm

4 Design and Implementation:

4.1 Functions Developed:

- FindImageMean Function: Gets image and filter size, and convolve image with mean filter and normalize it.
- FindImageStdDev Function: Applies standard deviation filter on image with 'stdfilt' function of matlab. Then it normalizes the image.
- ImageMaskFilter: Convolve image with given filter and normalizes the image.
- CoOccurenceMatricesScript Script: Calculating entropy image, and contrast, correlation, energy and homogeneity of the each pixel. Then it shows the results.It uses 'graycomatrix', 'graycoprops' functions of matlab.
- EnergyFiltersScript Script: This script separated into 2 part. First, convolving image channels, gray image and hue image with Laws' masks/filters and showing results. Later, it defines filter size and finding mean, absolute mean and standard deviation of the image channels, gray image and hue image. And then shows the result.

5 Experimental Results and Analysis

5.1 GLCM Results

In this section we've taken results for one build-in Matlab image with different orientations like 0, 45, 90, 135. Other parameters like windows size, numlevels, symmetric are let to default 9, 8 and false respectively.

Here are the results 6 images; original, entropy, contrast, correlation, energy and homogeneity.

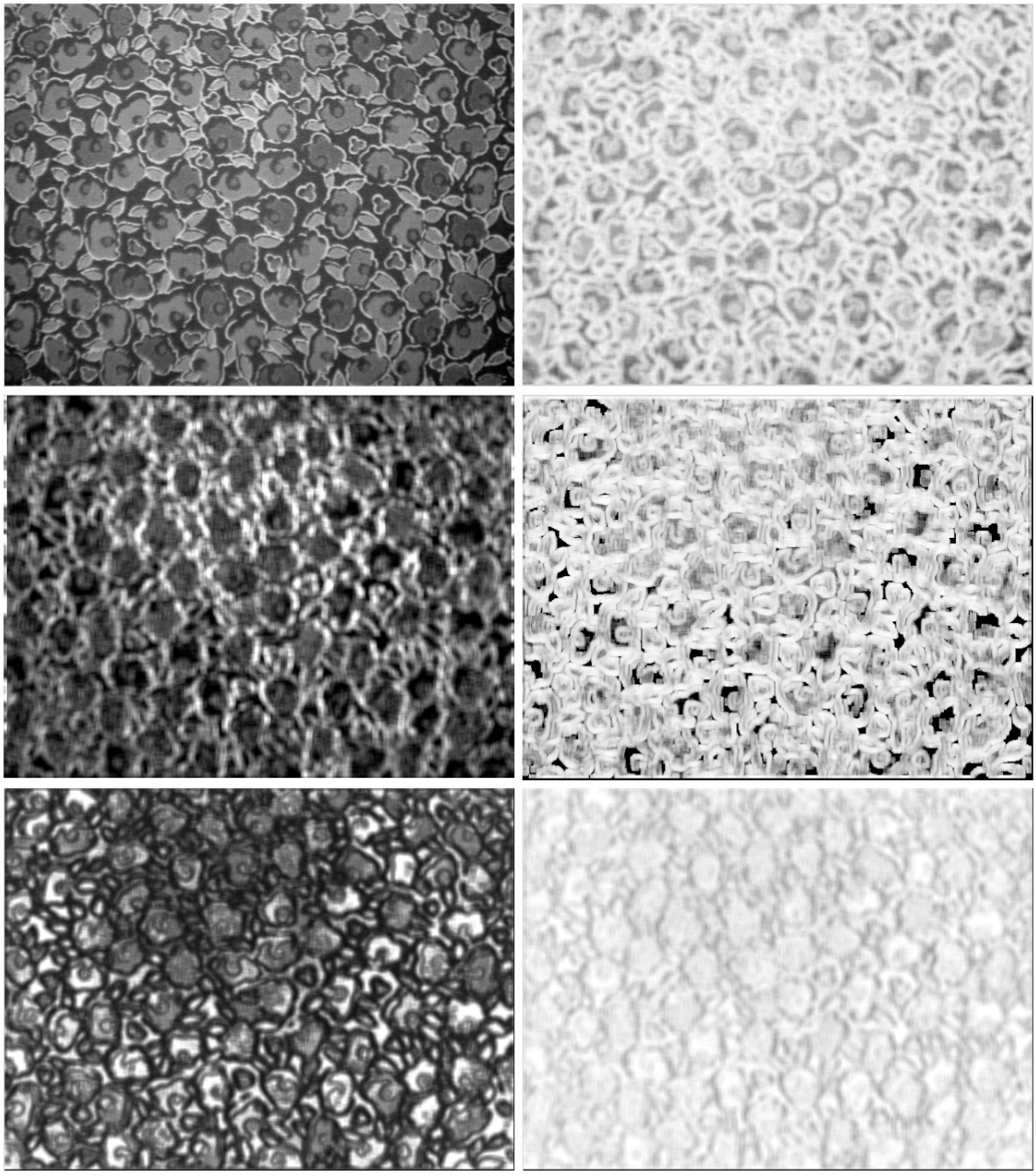

Figure 1: Results with 0 degree

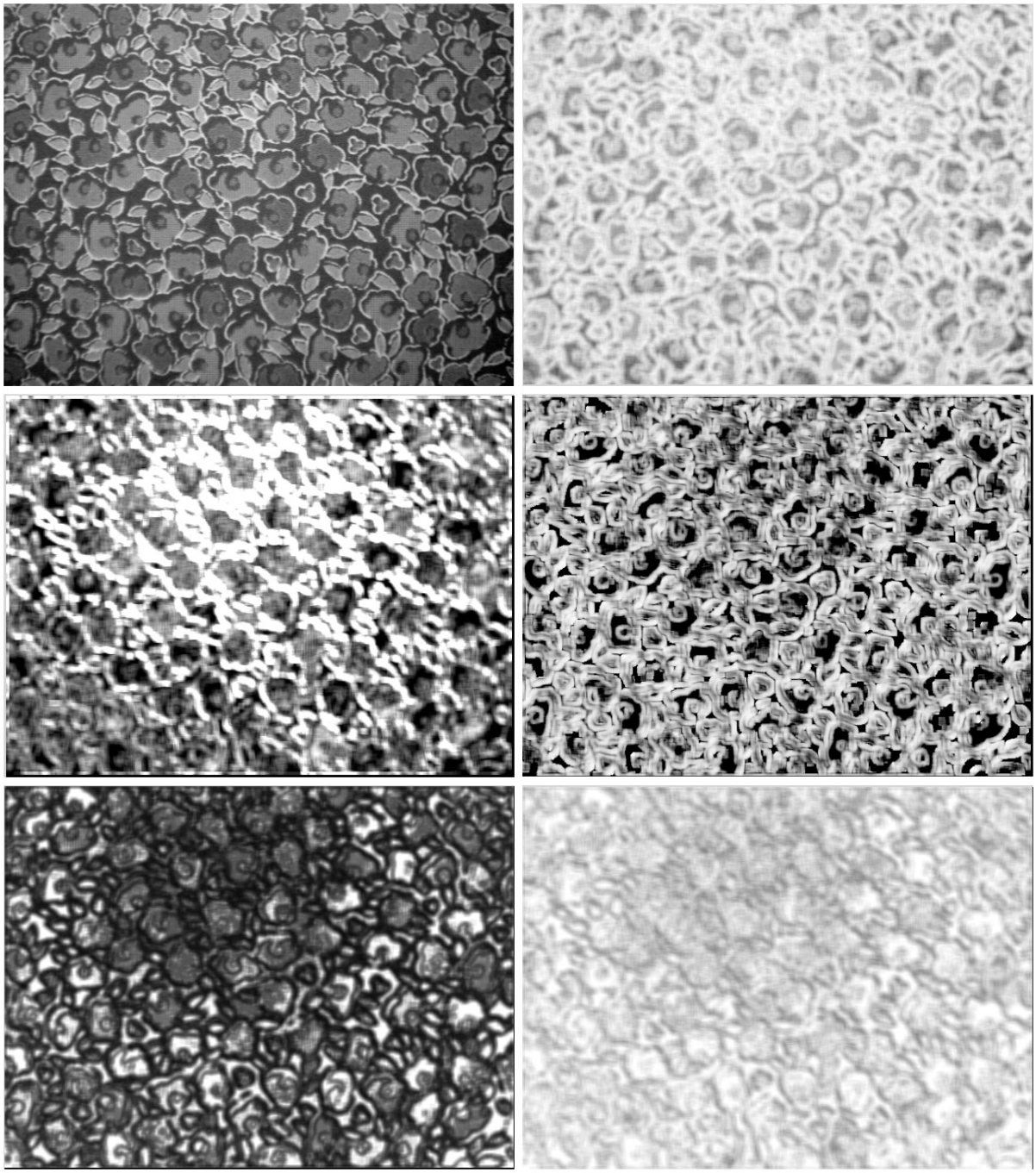

Figure 2: Results with 45 degree

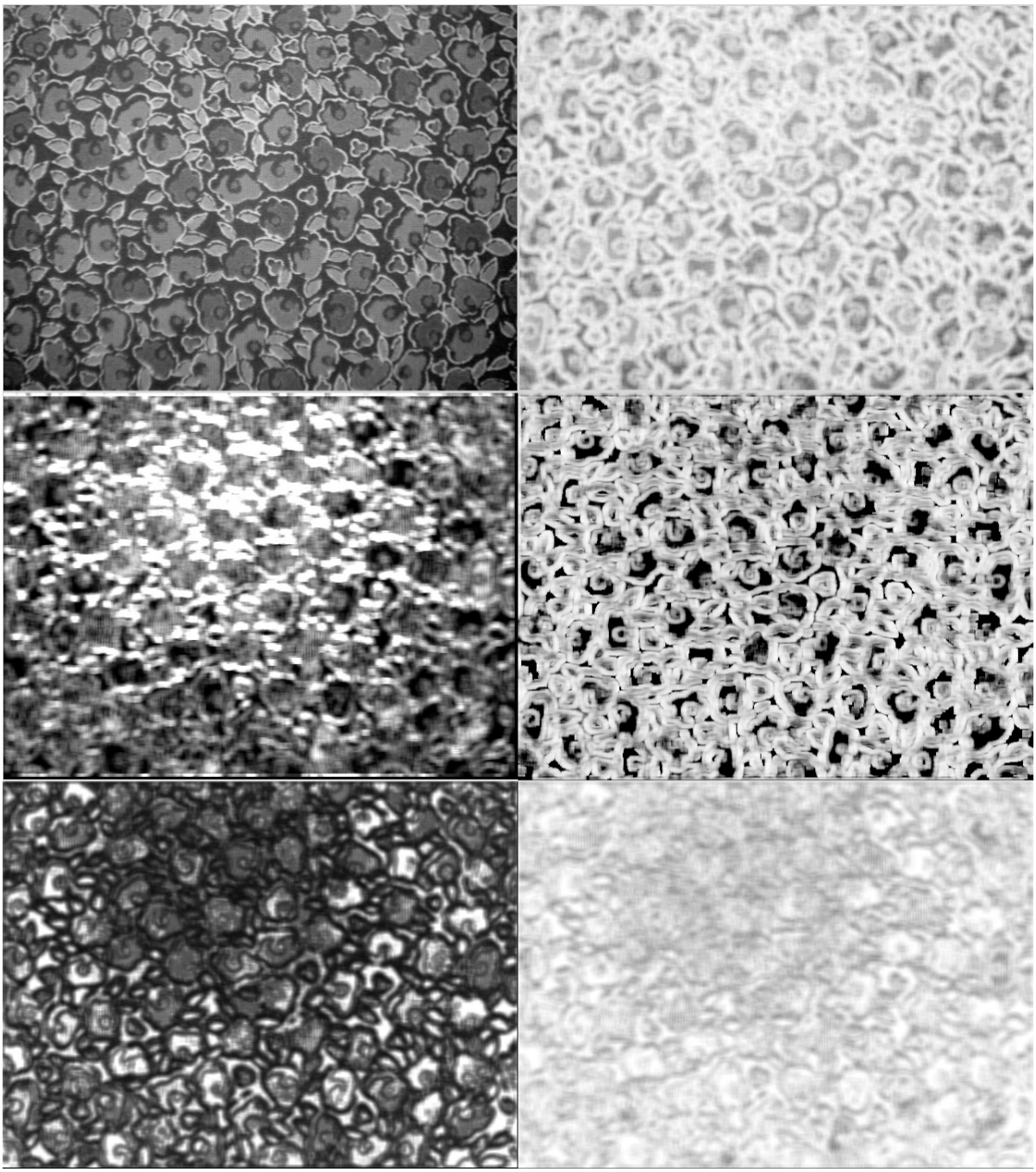

Figure 3: Results with 90 degree

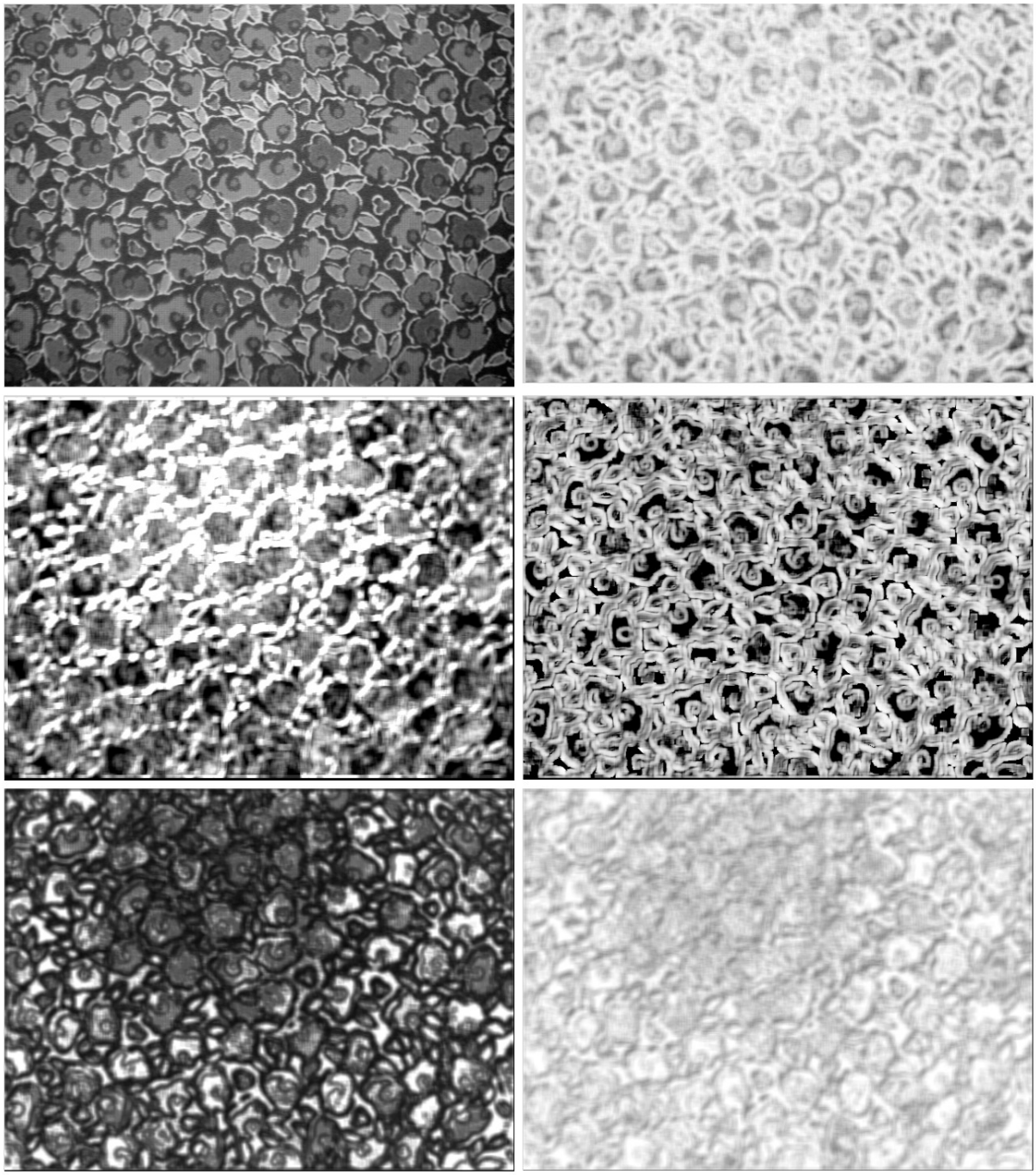

Figure 4: Results with 135 degree

5.2 Mean, Absolute Mean and Std.Dev. Results

In this section we calculated results with applying mean, absolute mean, standart deviation and normalization of the images of its channels, gray level and hue.

Here is the results:

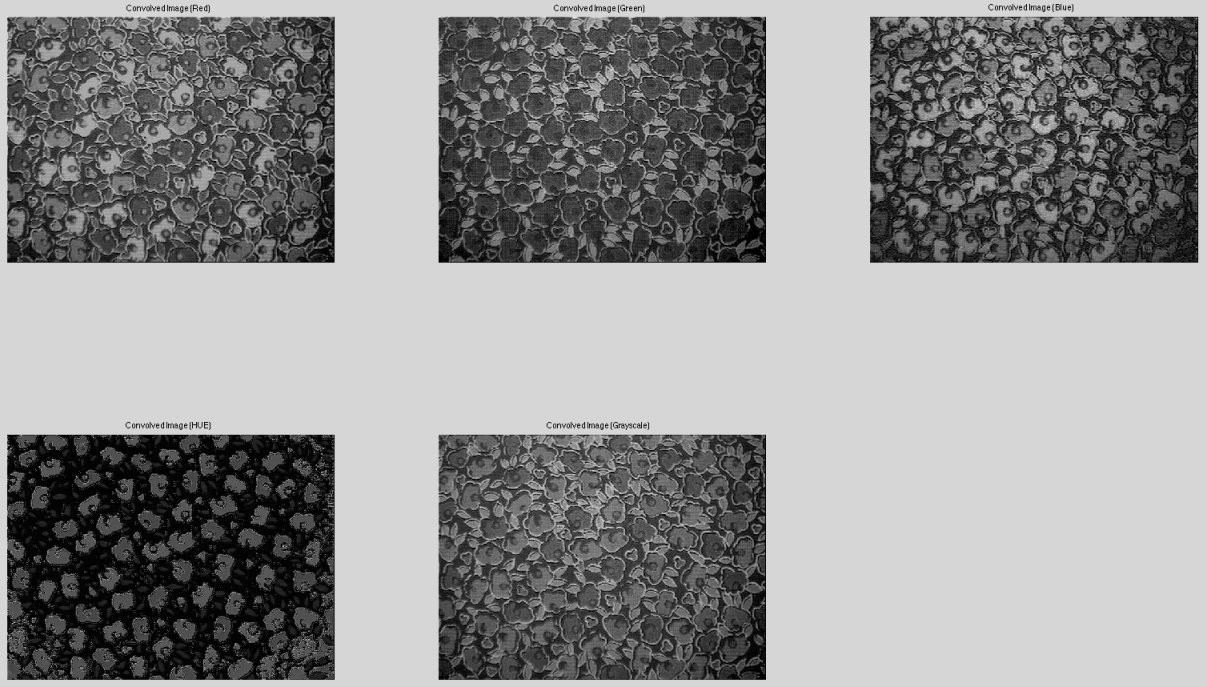

Figure 5: Laws Filter convolution of R, G, B channels, HUE and grayscale

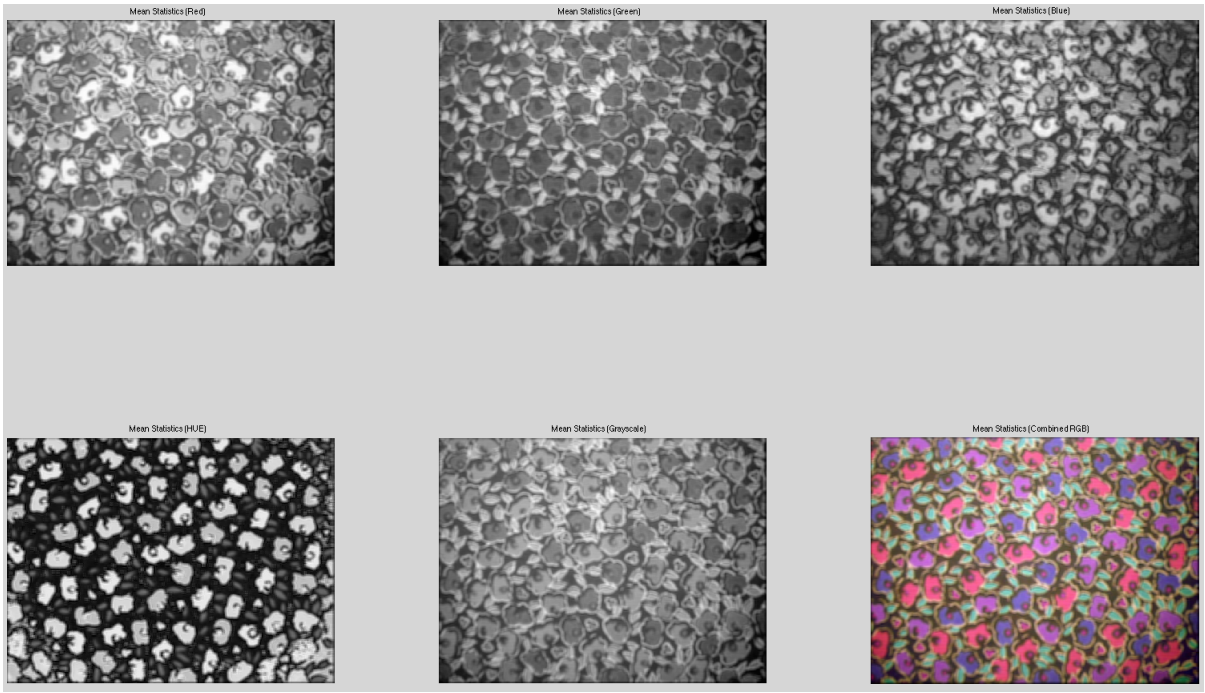

Figure 6: Mean of R, G, B channels, HUE and grayscale

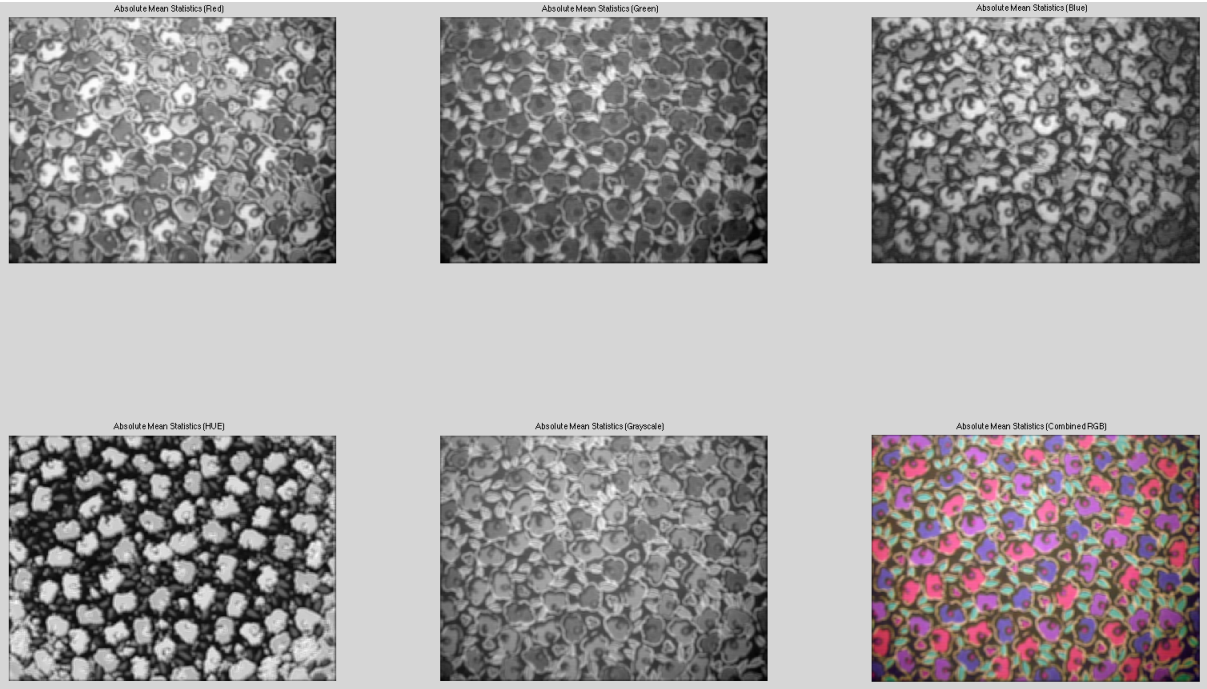

Figure 7: Absolute mean of R, G, B channels, HUE and grayscale

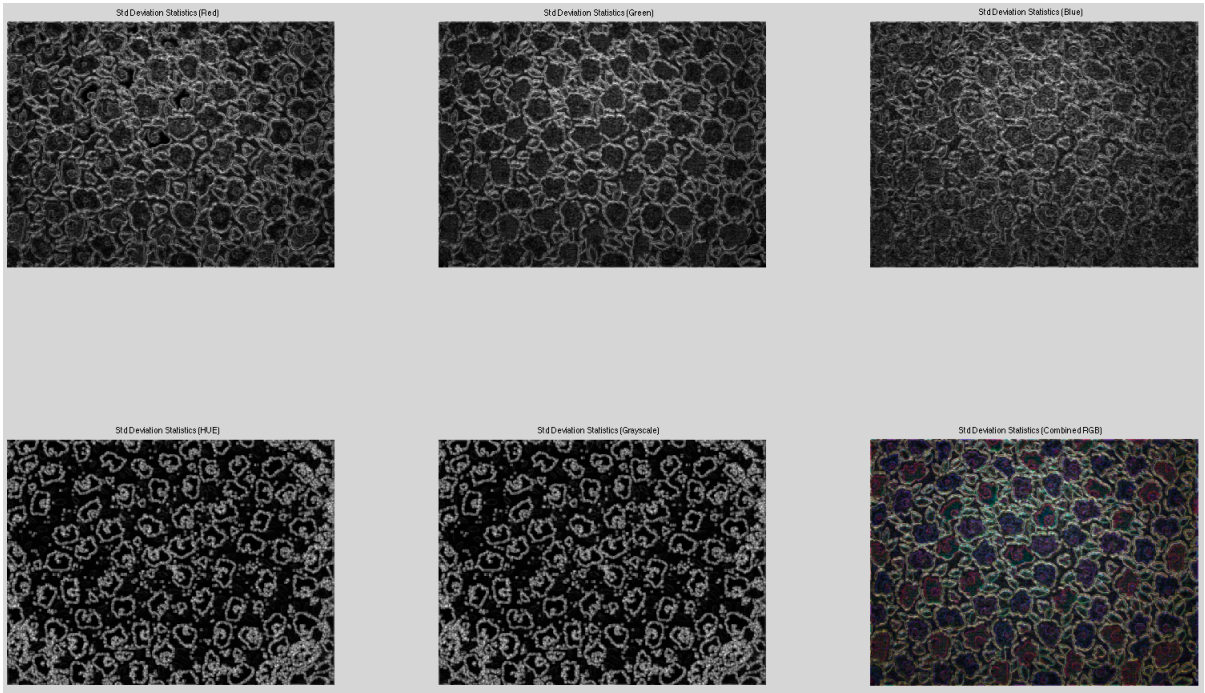

Figure 8: Standart deviation of R, G, B channels, HUE and grayscale

5.3 Solution:

References

[1] $http://www.fp.ucalgary.ca/mhallbey/what_is_texture.htm$

[2] Matlab Help

[3] $www.macs.hw.ac.uk/texturelab/files/publications/phds\_mscs/.../ch4.pdf$