CSE585/EE555: Digital Image Processing II

Computer Project # 4

Texture Segmentation

Mandar Parab, Amogh S. Adishesha, Lyuzhou Zhuang

Date: 4/7/2017

A. Objectives

Learn to segment different textures in an image using Gabor filter. Implement Gabor filter in Matlab.

B. Methods

In this project, we apply Gabor filter to multiple images to segment different textures in these images. We use the same formula but with different parameter values to construct Gabor filters for different images. Figure 2.1 is the MATLAB program flowchart. (Note: Run main.m and it will automatically call subfuntions to perform any operations required.)

1. Read in original image(s) I(x,y)

2. Compute circularly-symmetric Gaussian g(x,y) with parameter values provided in the problem description.
   We compute g(x) and g(y) separately, then combine them together, g(x,y) = g(x)*g(y), where

$$g(x,y) = \frac{1}{2\pi\sigma^2}\exp\left\{\frac{-(x^2+y^2)}{2\sigma^2}\right\}$$

3. Compute GEF h(x,y) with parameter values provided in the problem description.

$$h(x,y) = g(x,y)\cdot\exp\left[j2\pi F(x\cos\theta + y\sin\theta)\right] = g(x,y)\cdot\exp\left[j2\pi(Ux+Vy)\right]$$

   Since h(x,y) is separable in x and y, namely, h(x,y) = $h_1(x) \cdot h_2(y)$, we compute $h_1(x)$ and $h_2(y)$ separately and apply them to (convolute with) input image successively.

4. Apply Gabor filter it input image(s) I(x,y) following instructions on L18-4 and L18-6

   1. $i_1(x,y) = i(x,y) * h_1(x)$ convolution in x

      $= i(x,y) * g_1(x)\exp\{j2\pi Fx\cos(\theta)\}$

   2. $i_2(x,y) = i_1(x,y) * h_2(y)$ convolution in y

   3. $m(x,y) = |i_2(x,y)|$

   where

   $$i_1(x,y) = \sum_{x'=x-3\sigma}^{x+3\sigma} i(x-x',y)\hat{h}_1(x')$$

   $i_2(x,y)$ is computed similarly.

   We then get the output image as m(x,y)

5. Compute smoothing filter g'(x,y)

   Same as in step 2 where we compute g(x,y) only that we use a different σ to compute g'(x,y).

6. Apply smoothing filter g'(x,y) to m(x,y) (if required)

   *m'(x,y) = m(x,y) * g'(x,y)*

7. Segment different textures and display results

   (Manually) choose a threshold that best segments different textures base on m'(x,y). Superimpose the image-segmentation result on the original image.

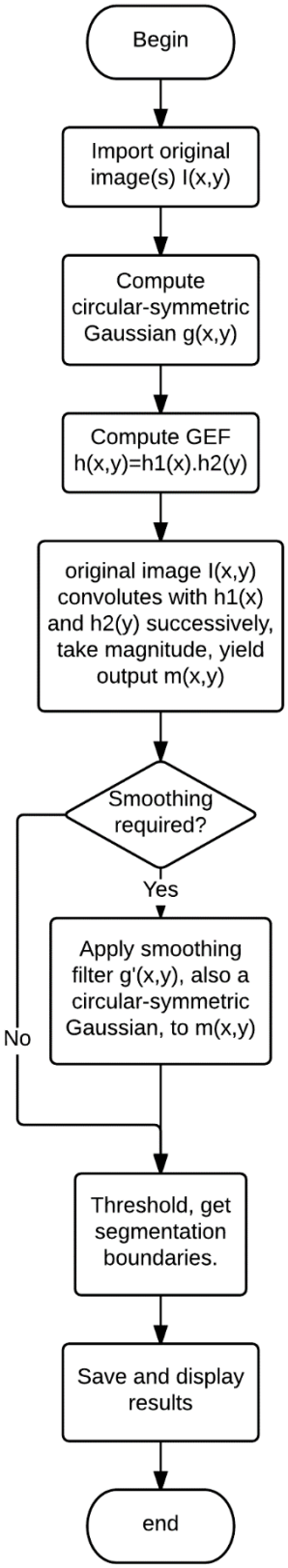

Figure 2.1 program flowchart

C. Results

Problem 1

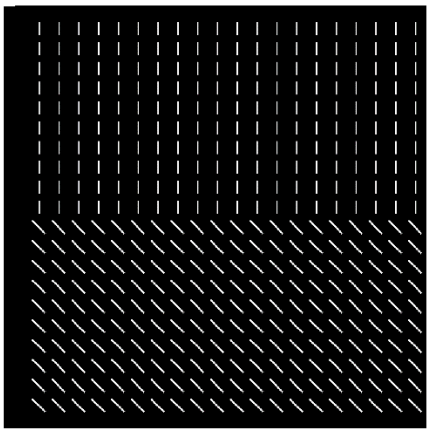

Fig 3.1 original image

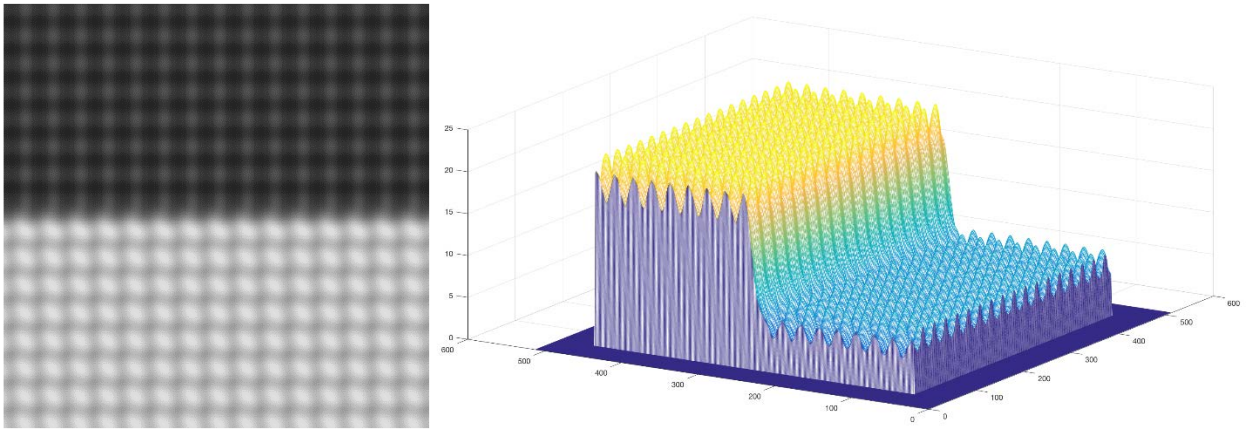

Fig 3.2 m(x,y), result of applying Gabor filter with F=0.059, θ=135, σ=8

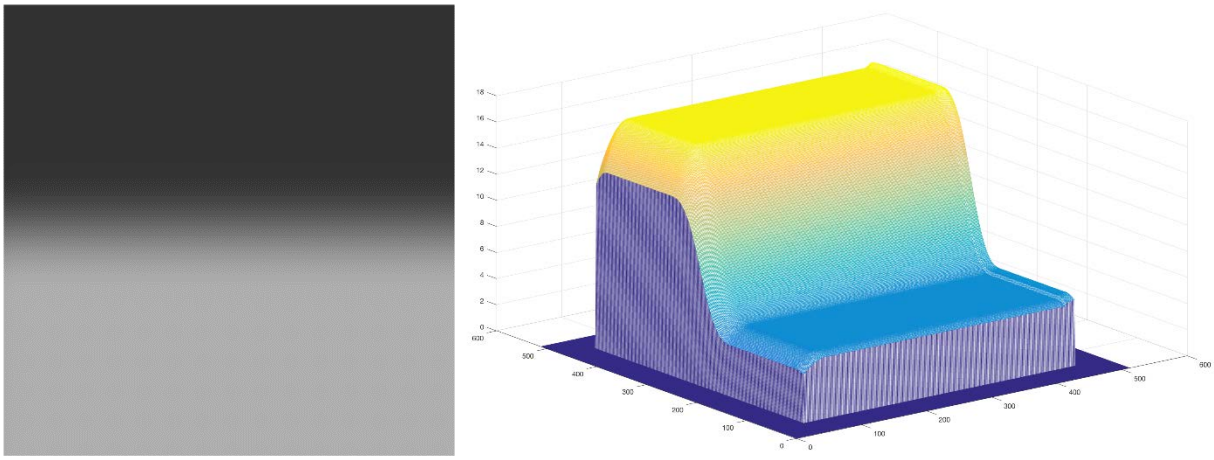

Fig 3.3 m'(x,y), result after applying smoothing filter with σ = 24

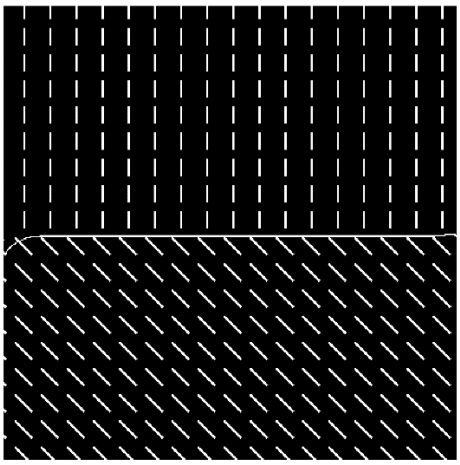

Fig 3.4 segmented image with threshold = 12

Problem 2

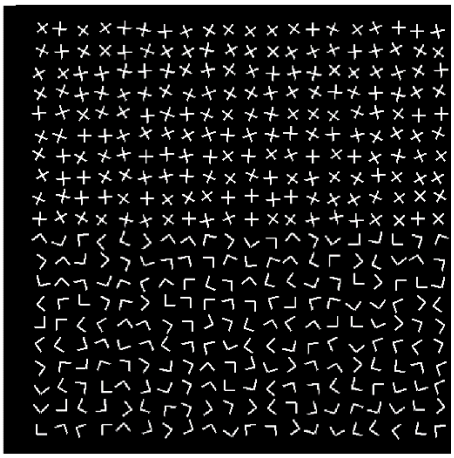

Fig 3.5 original image

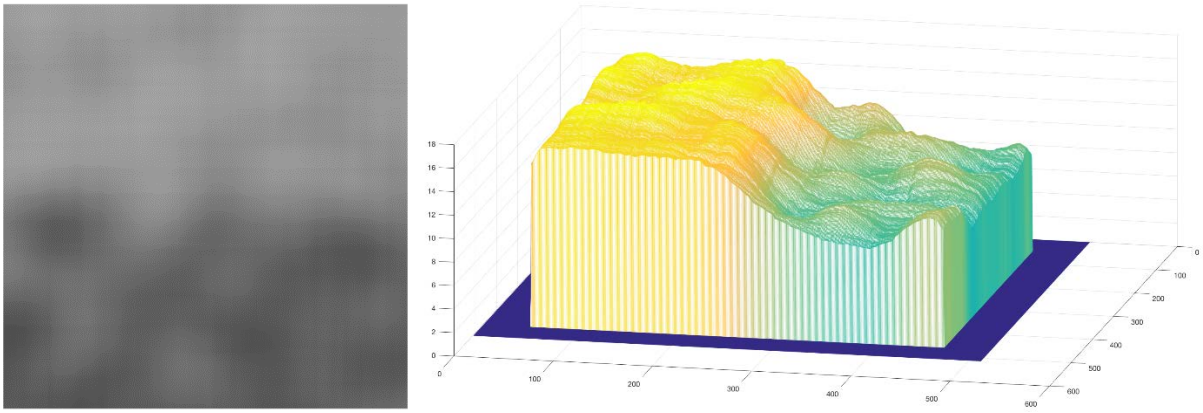

Fig 3.6 m(x,y), result of applying Gabor filter with F=0.042, θ=0, σ=24

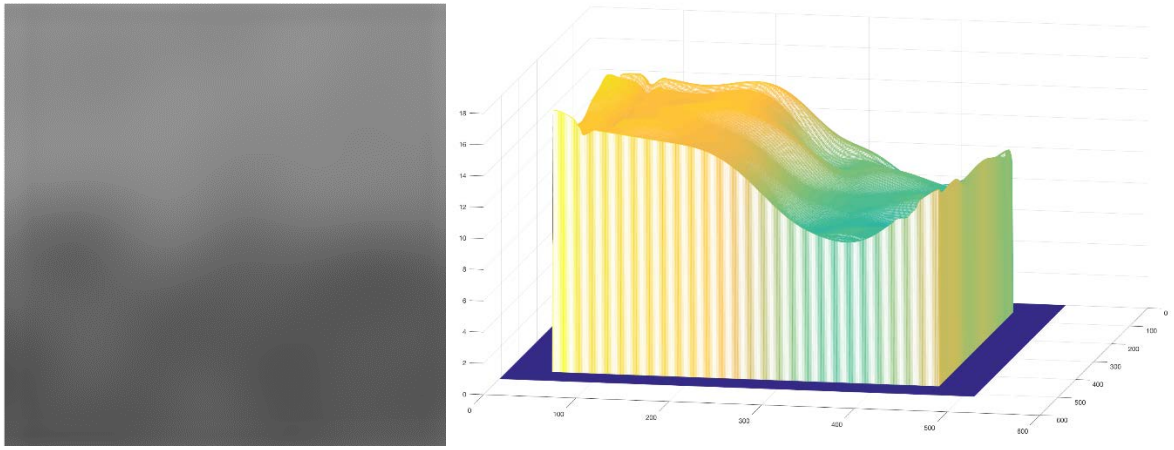

Fig 3.7 m'(x,y), result after applying smoothing filter with σ = 24

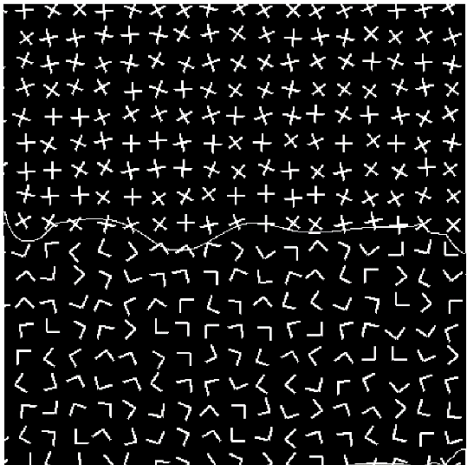

Fig 3.8 segmented image with threshold = 12

Problem 3

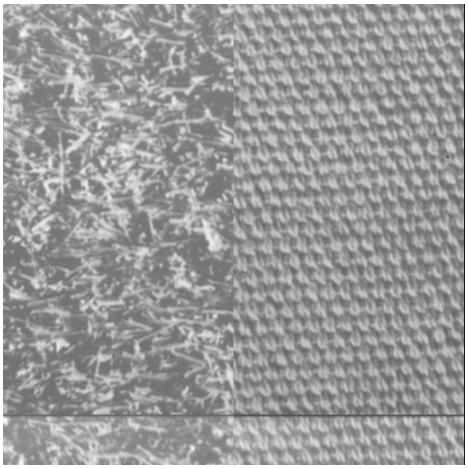

Fig 3.9 Original image

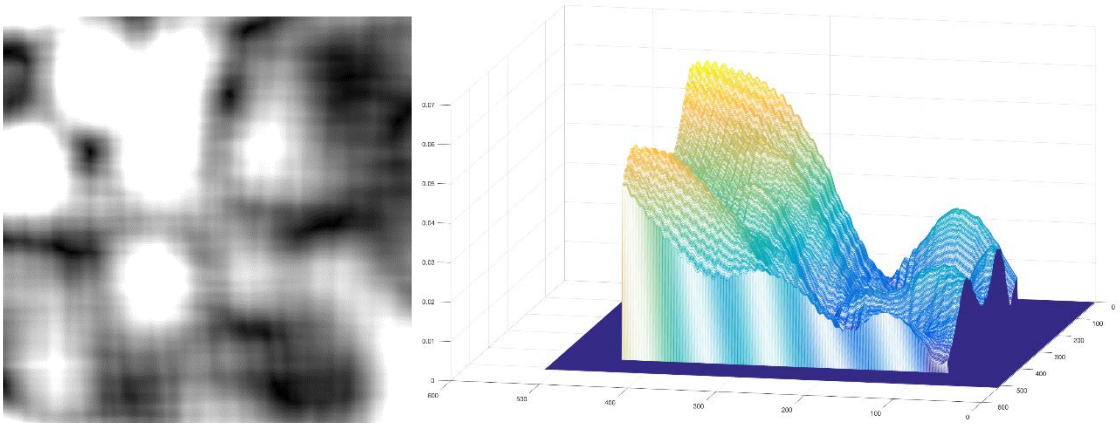

Fig 3.10 m(x,y), result after applying Gabor filter with F=0.063, θ=-45, σ=36

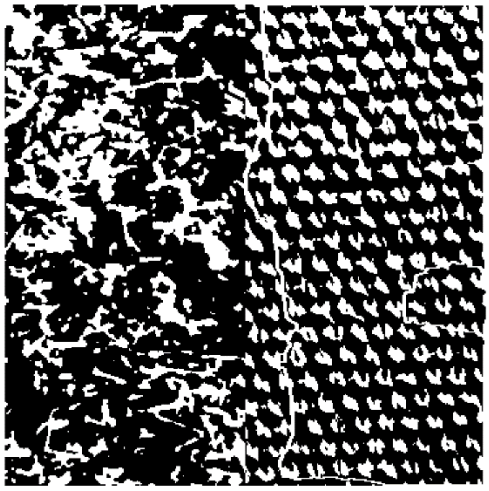

Fig 3.11 segmented image with threshold = 0.025

Comments:

The results show that Gabor filter works reasonably well with respect to segmenting textures. Problem 1 works the best, Problem 3 the worst. It seems that the more distinct and organized the textons, the better the segmentation result. Also from the 3D plots, we can tell that chosen the right scope, smoothing helps a lot and reduce error when we simply use a threshold to segment 2 textures. But smoothing may also blur edges and introduce new error.

In this project, we did not zero-pad image boundaries, so with every filter applied, the image gets smaller. We may need to take this effect into account in future implementations.

D. Conclusions

With the help of smoothing filter, Gabor filter works well detecting different textures. Note that parameter values need to be chosen carefully to achieve desirable results. Another way to segment textures, instead of using a single Gabor filter, is to design different filters to detect different textures. Besides, we are now setting parameter values manually. We can introduce optimization algorithms in the future to update these values every time it works on a new image.