**CSE585/EE555:  Digital Image Processing II**

**Computer Project # 4:**

**Nonlinear Filtering and Anisotropic Diffusion**

**Yanxi Yang, Jiuchao Yin, Hongjie Liu**

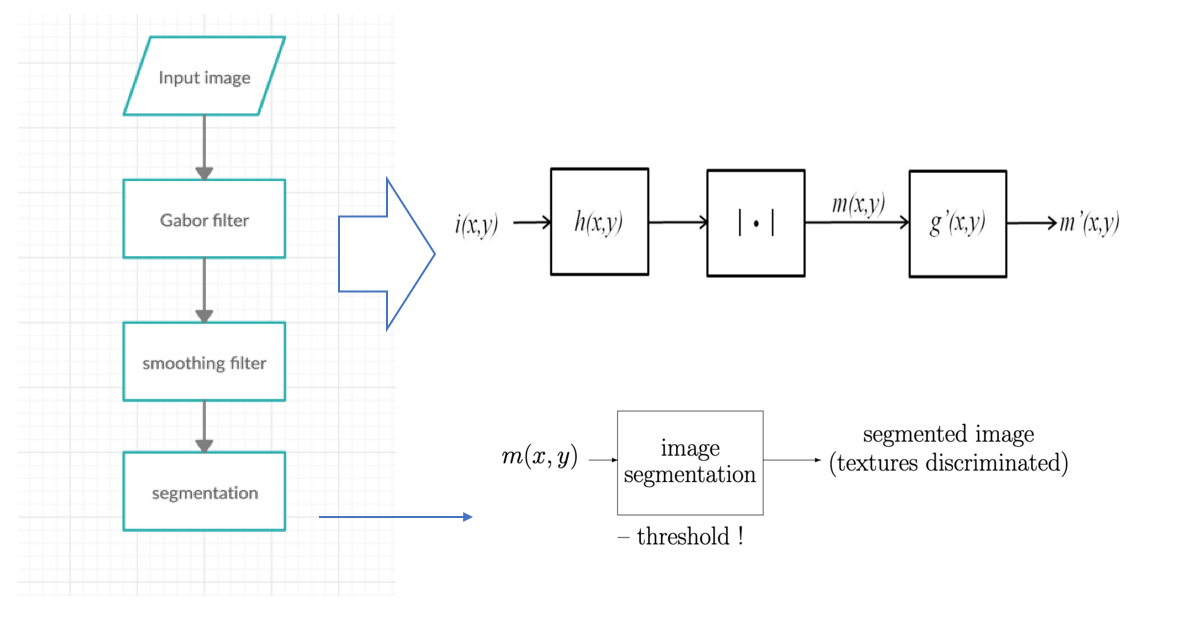
**Date: 04/06/2020**

1. **Objectives**

* To classify and segment bipartite texture regions in an image.
* To understand the principles of Gabor filter and how to design a single Gabor filter on textured image to accomplish optimal segmentation.
* To learn the algorithm of Gaussian part (low pass filter) and complex sinusoid part of Gabor elementary function (GEF).
* To investigate the parameters specify the Gabor filter and smoothing filter.
* To get familiar with image 3-D plot from texture analysis from Gabor filter and smoothing filter.
* To extract boundaries between major textures regions and to explore the criteria of defining a good segmentation of texture.

1. **Methods**

**Flow chart**



**Figure1. flow chart of Gabor filter texture segmentation**

**Code structure:**

**main.m:** the main code runs the 4 different tasks based on 4 images with various parameters settings, and output the grayscale images and 3D plot with m(x,y), m’(x,y) and final segmented results.

**support functions:**

* g.m: the Gaussian part of GEF This function is used to compute the circularly-symmetric Gaussian of one pixel. The inputs are the value of sigma and the x/y coordinate of the pixel. The input includes only x or only y since we compute convolution in x and y separately in main function. The output is the circularly-symmetric Gaussian value of that pixel. This function does not call any other functions.
* hx.m: GEF of h(x) Since we compute GEF in x and y separately in main function, is separated into and . This function is used to compute the GEF in x, i.e. . The inputs are F, theta, sigma, and x coordinate of the pixel. The output is the computation result of x coordinate. This function does not call any other functions.
* hy.m: GEF of h(y) This function is the same as hx.m. The only difference between them is that the inputs of this function are F, theta, sigma, and y coordinate of the pixel, not x. This function is used to compute the GEF in y, i.e. and does not call any other functions.
* segment.m: This function is used to do segmentation with discriminative threshold of each texture for classification. The inputs are results after Gabor filter ( or ), the original image, sigma, and threshold. The output is the visualized segmentation result. This function does not call any other functions.

1. **Algorithms**

To implement an optimal Gabor filter for texture segmentation, by looking at the output of m(x,y) that if it is a noisy step function for better discrimination and classification. Then we shall determine whether the GEF’s parameters are properly set or not.

**Algorithms are as following:**

By applying the Gabor filter:

m (x,y) = [I(x,y) \*\* h(x,y)]

where I denotes the input image, h is a GEF :

h(x,y) = g(x, y) exp [j2πF(xcosθ + ysinθ)] = g(x, y) exp [j2π (Ux + Vy)]

where, θ is the orientation of sinusoid.

and g is a circularly-symmetric Gaussian:

g(x,y) = exp {}

The assumption of this project is that Φ=0, so we can implement the GEF separably for x and y:

h(x,y) = h1(x) h2(y)

Thus, the GEF can be processed through three steps:

* i1(x,y) = i(x,y) \* h1(x)

=

* i2 (x,y) = i1(x,y) \* h2(y)

=

* m(x,y) = | i2 (x,y)|

Thus, Gabor filter has width of (4).

To get easier for the segmentation, apply the smoothing filter:

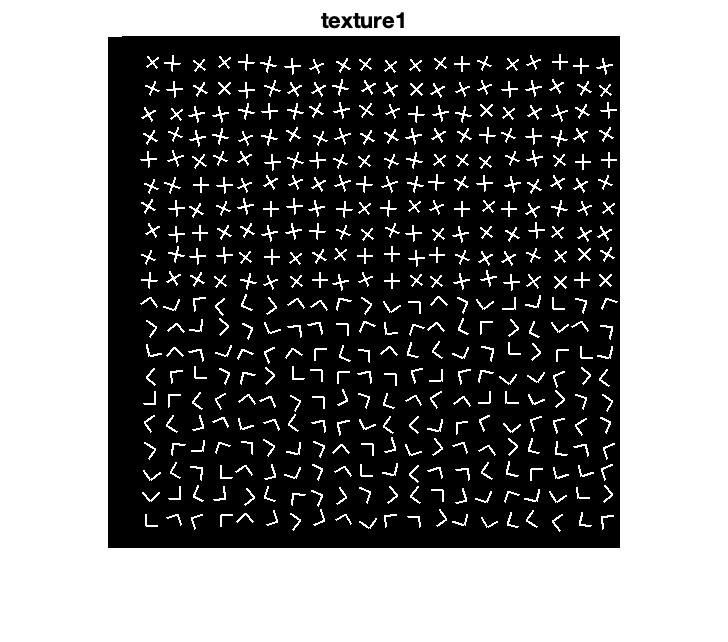
m’(x,y) = m(x,y) \* g’(x,y)

where g′ is another circular-symmetric Gaussian using a different .

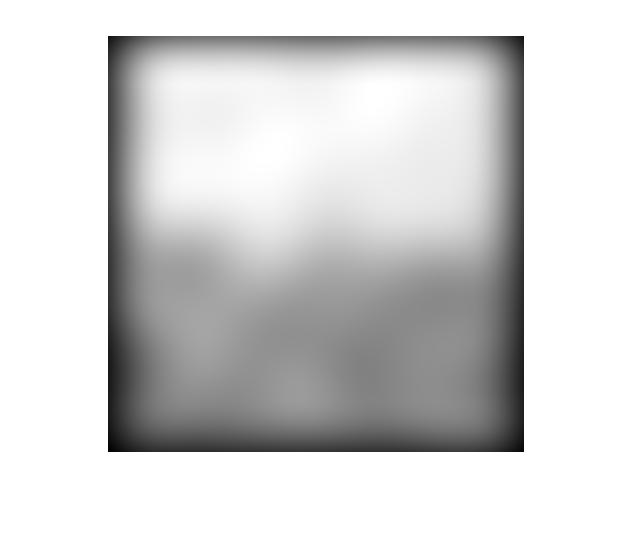
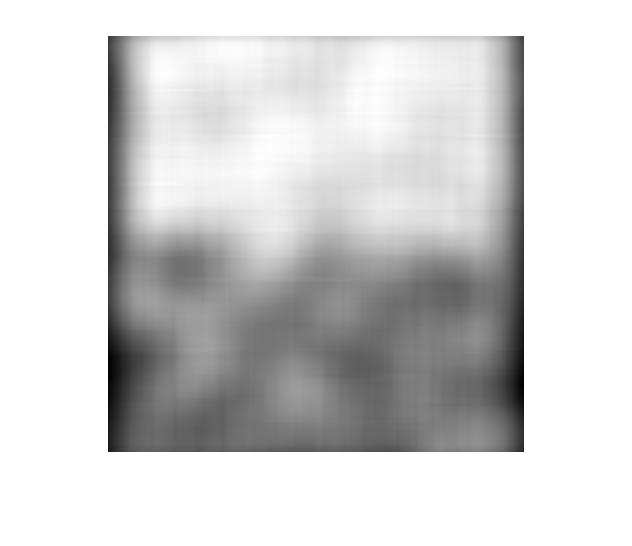
1. **Results**

Since that values near the outer perimeter of the image cannot be completely processed, thus, when displaying results, we have zeroed out unprocessed perimeter areas.

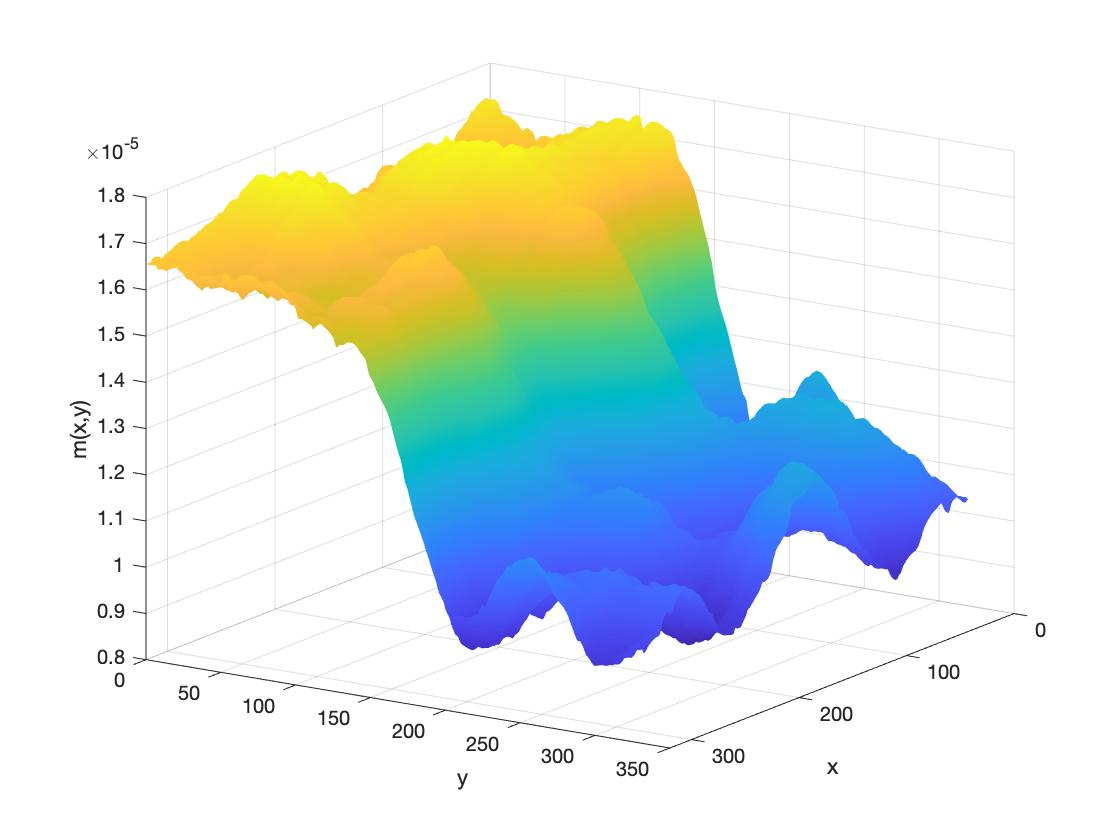
Image ‘texture1’:



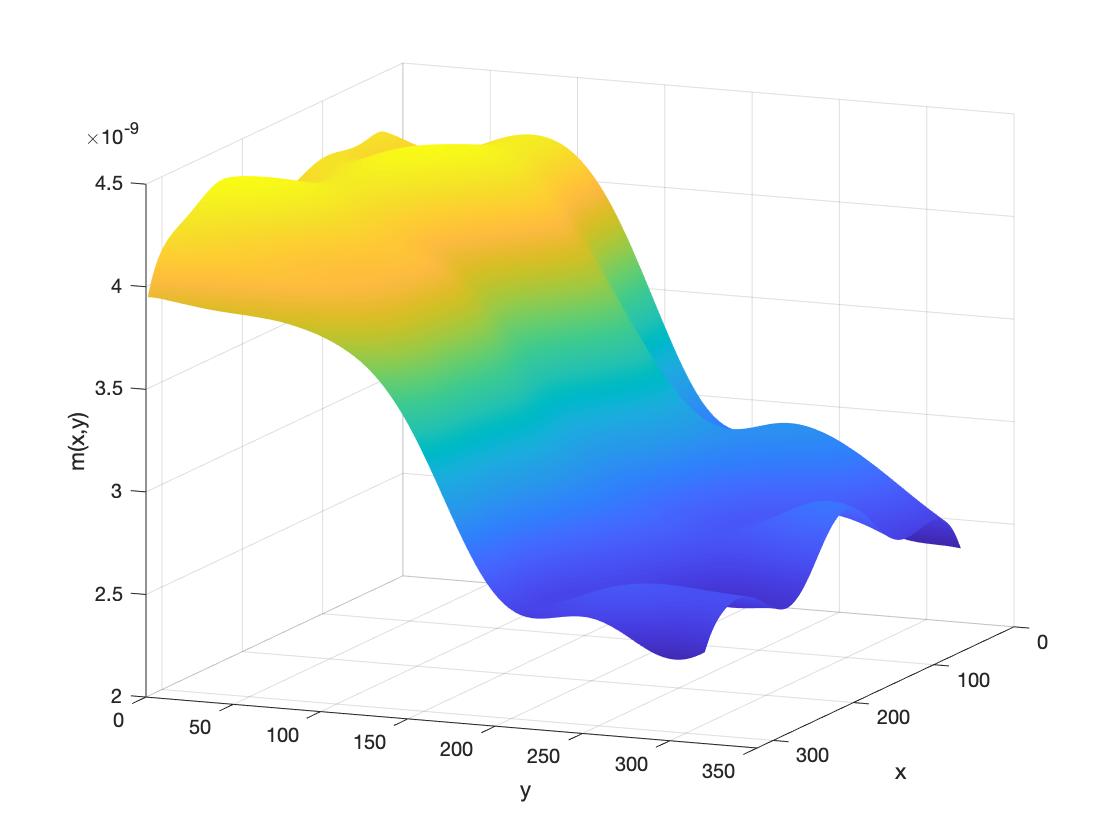
**Figure 2. binary image of ‘texture1’**



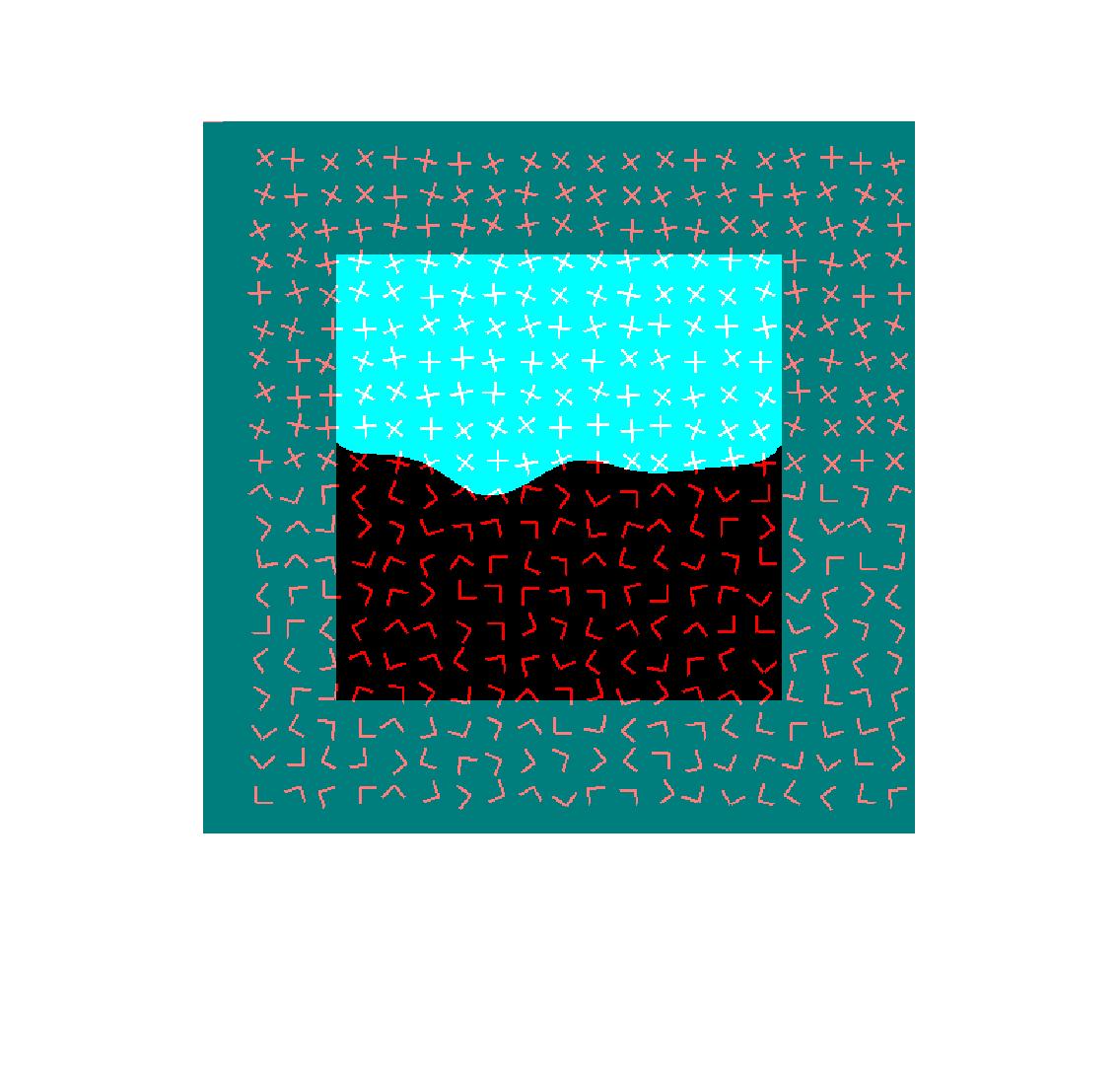
**Figure 3. grayscale image of Gabor filter (left) and smoothing filter (right)**



**Figure 4. 3D plot of ‘texture 1’ after Gabor filter**



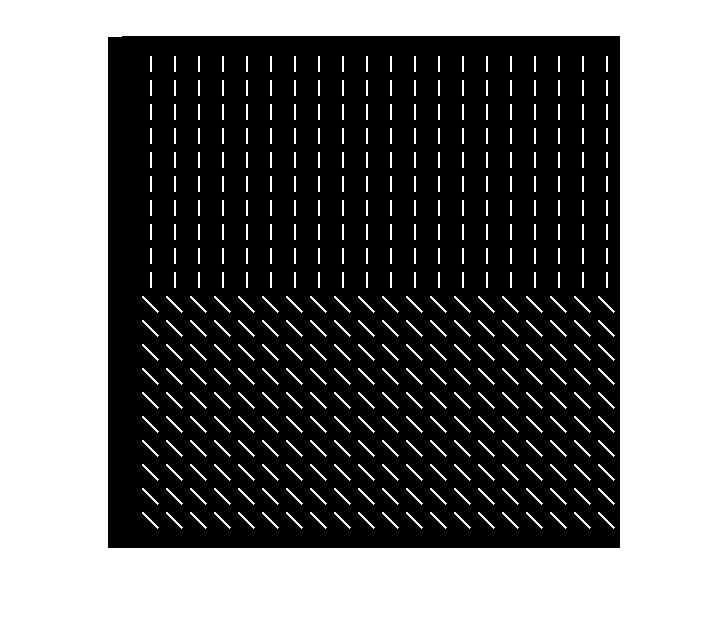
**Figure 5. 3D plot of ‘texture 1’ after Gabor filter and smoothing filter**



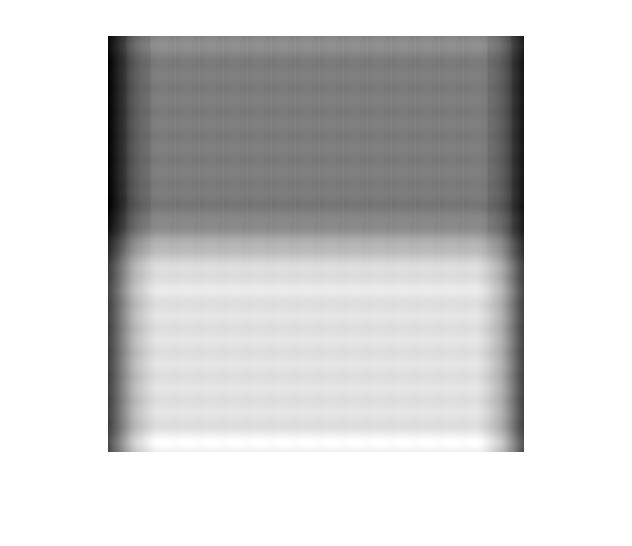
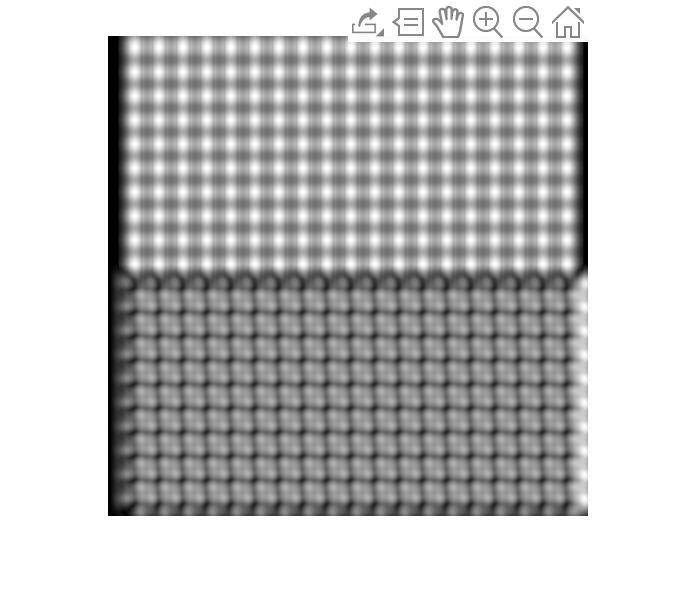
**Figure 6. superimposition of texture segmentation of ‘texture1’**

Observation:

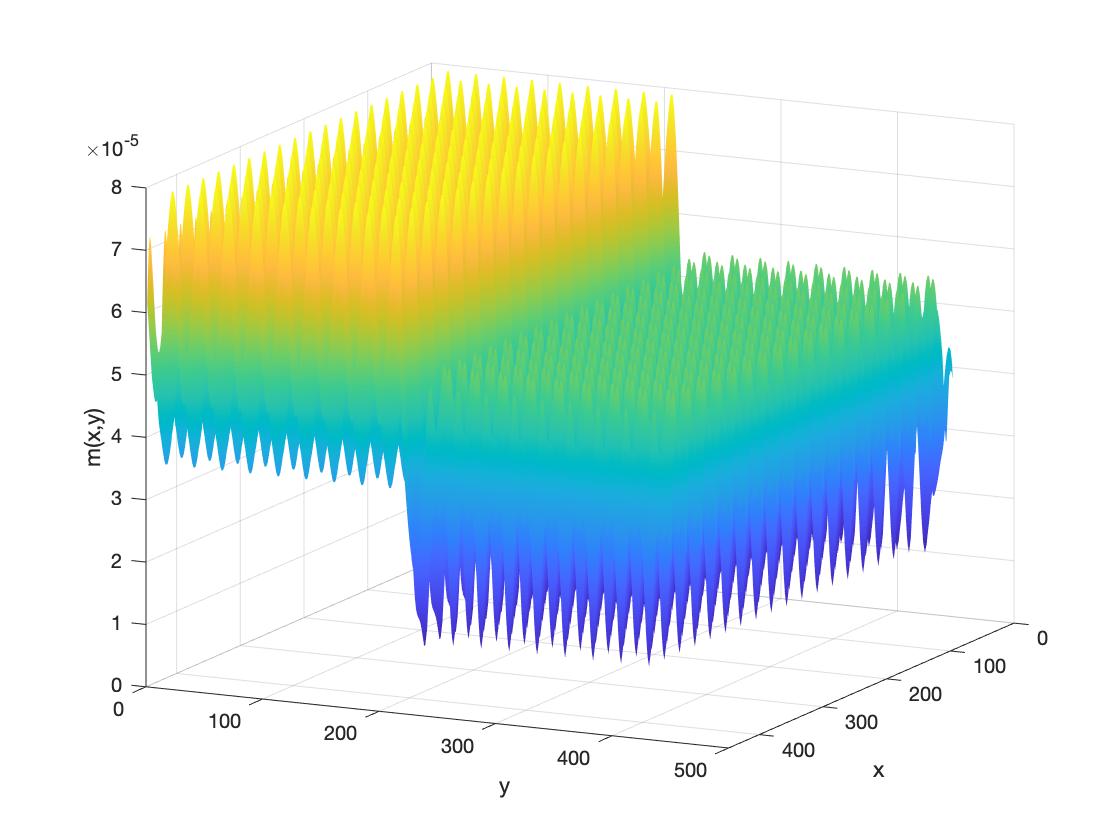
image ‘texture2’:



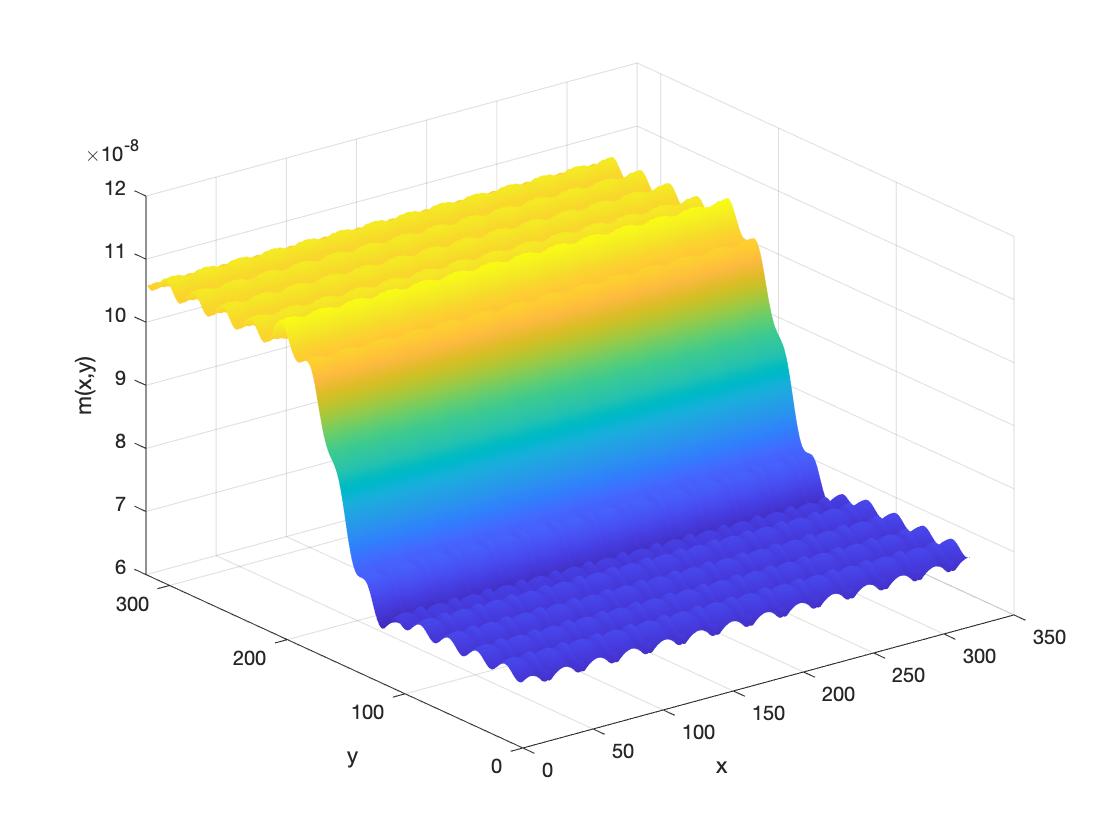
**Figure 7. binary image of ‘texture 2’**



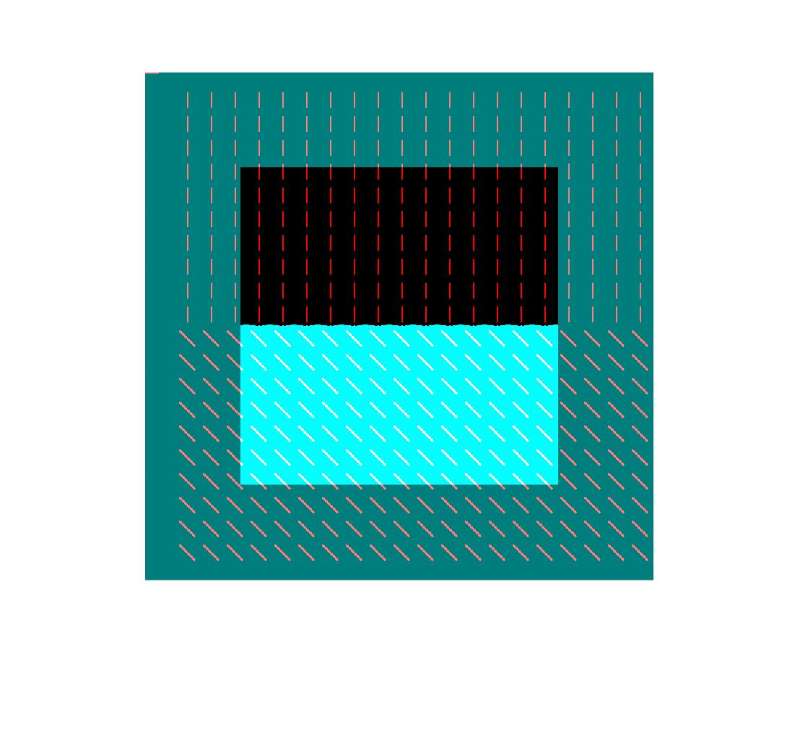
**Figure 8. grayscale image of Gabor filter (left) and smoothing filter (right)**



**Figure 9. 3D plot of ‘texture 2’ after Gabor filter**



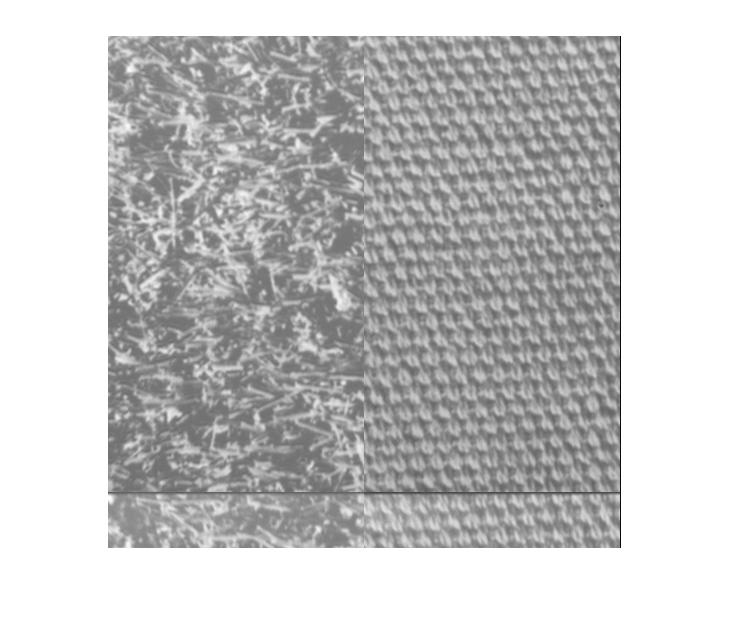
**Figure 10. 3D plot of ‘texture 2’ after Gabor filter and smoothing filter**



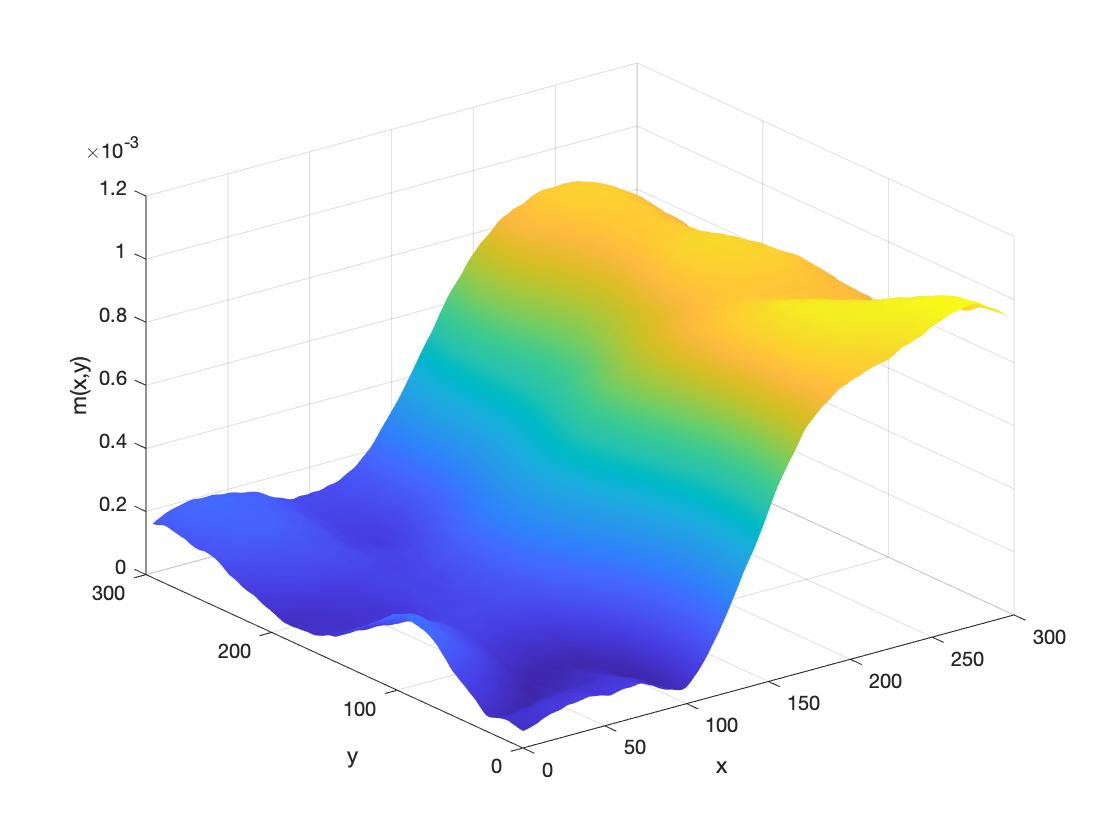
**Figure 11. superimposition of texture segmentation of ‘texture2’**

Observation:

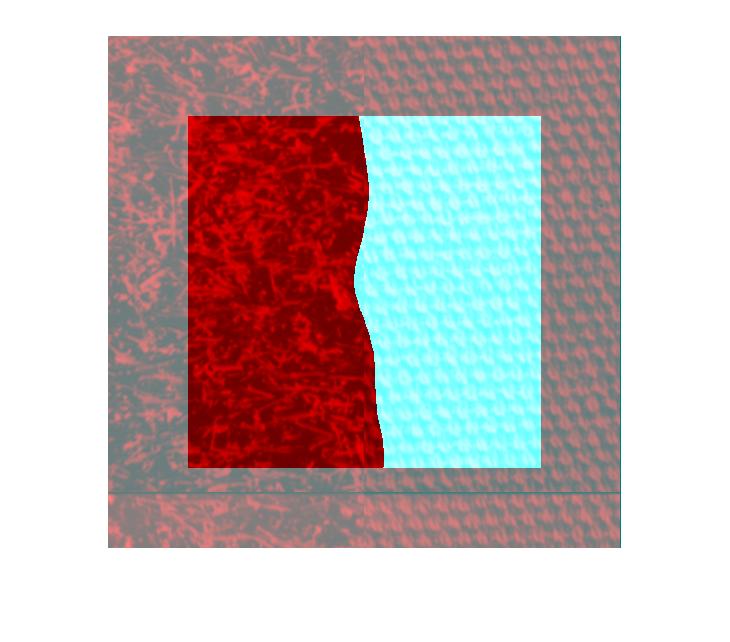
**image ‘d9d77’**:

**Figure 12. grayscale image of ‘d9d77’ (left) and after Gabor filter (right)**



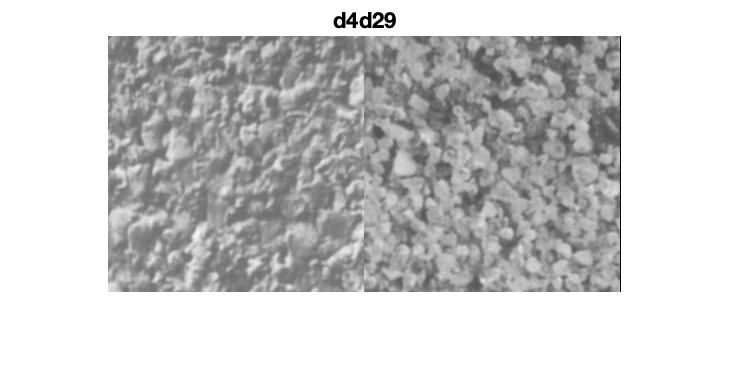
**Figure 13. 3D plot of ‘d9d77’ after Gabor filter**



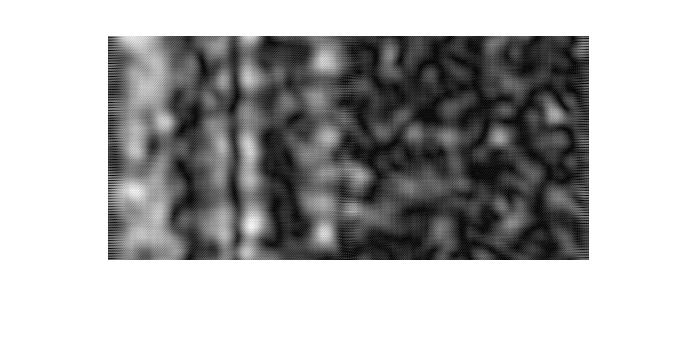
**Figure 14. superimposition of texture segmentation of ‘d9d77’**

Observation:

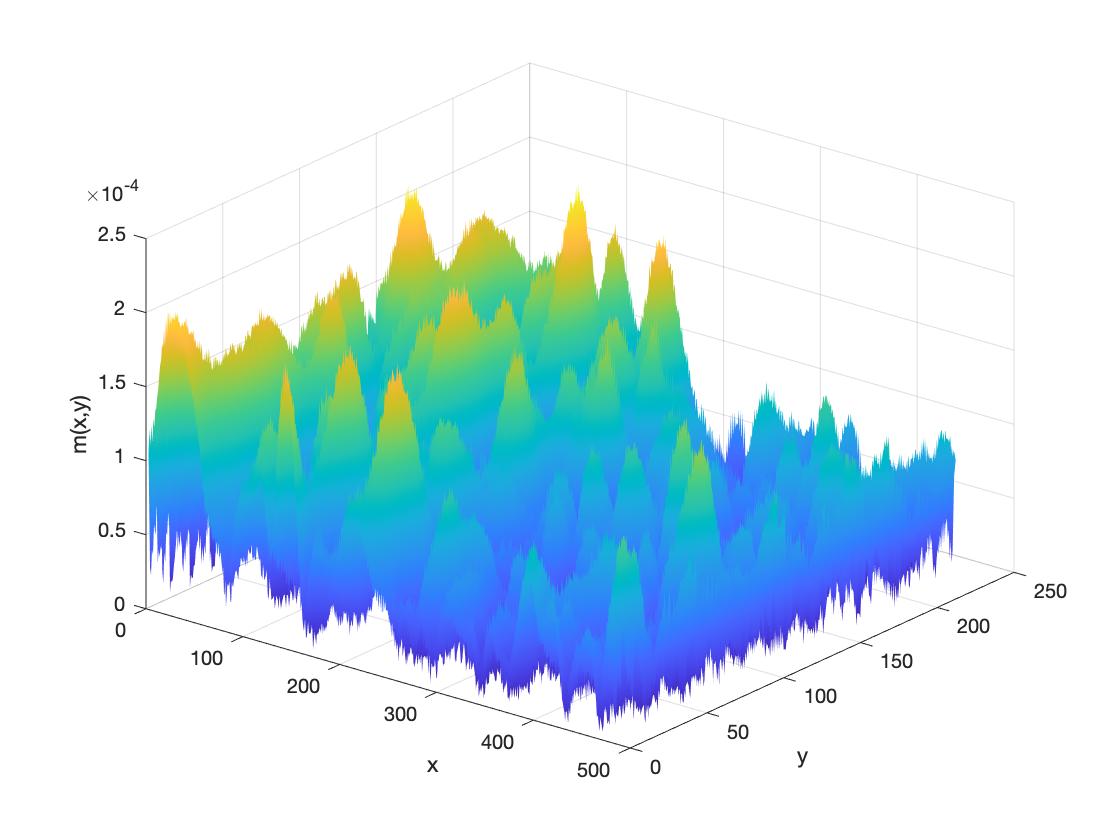
image4:



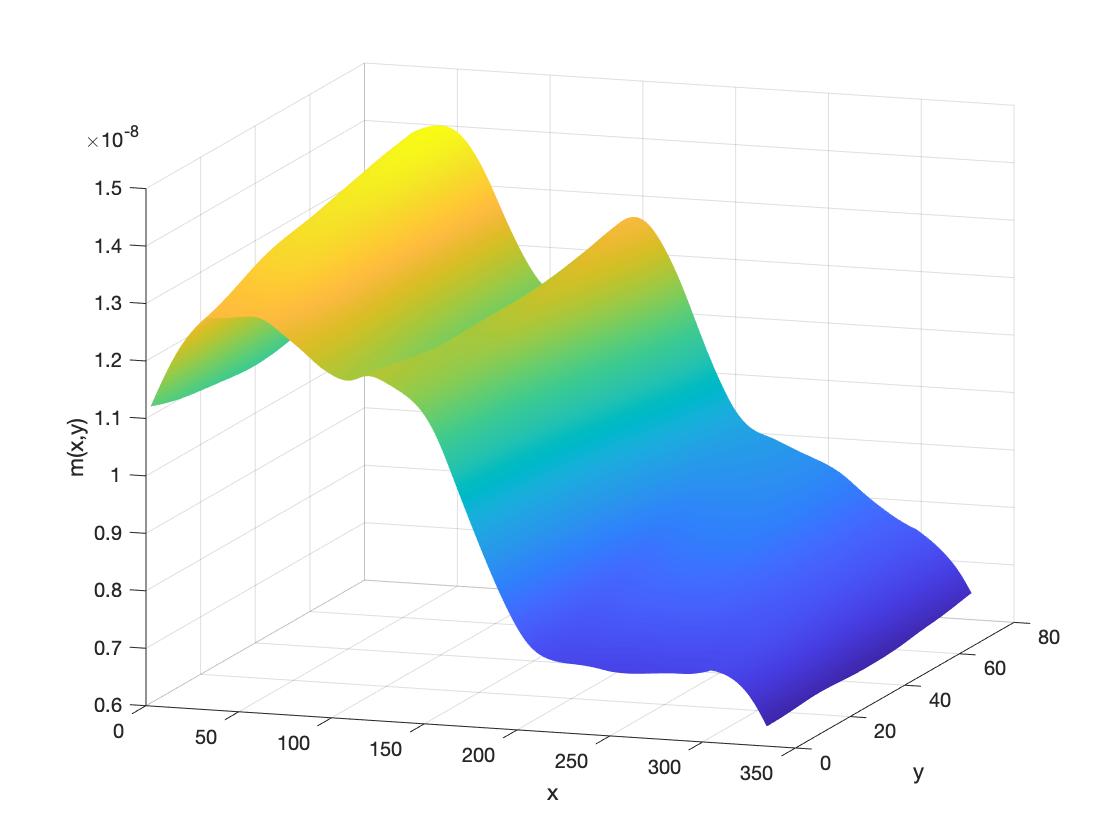
**Figure 15. binary image of ‘d4d29’**



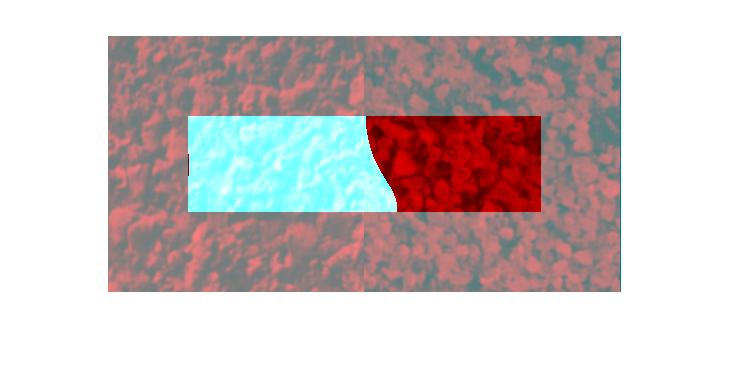
**Figure 16. grayscale image of Gabor filter (left) and smoothing filter (right) of ‘d4d29’**



**Figure 17. 3D plot of ‘d4d29’ after Gabor filter**



**Figure 18. 3D plot of ‘d4d29’ after Gabor filter and smoothing filter**



**Figure 20. superimposition of texture segmentation of ‘d4d29’**

Observation:

1. **Conclusion**

In conclusion, the project

The Gabor filter has selectivity for orientation, spectral bandwidth and spatial extent.