CSE585/EE555: Digital Image Processing II

Computer Project # 4:

Texture Segmentation

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 specifying 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.

2. Methods

2.1 Flow chart

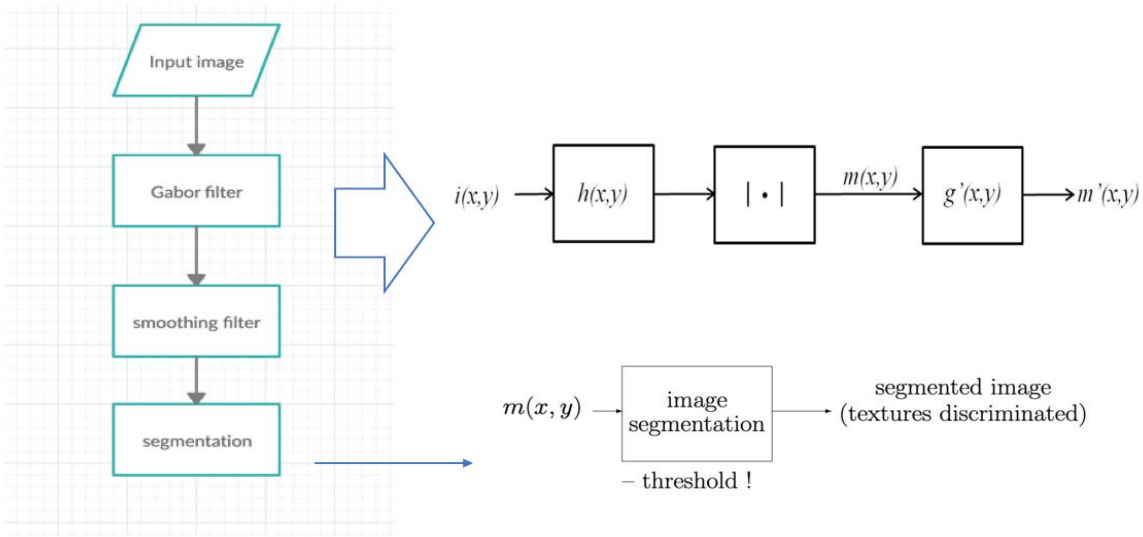

Figure1. flow chart of Gabor filter texture segmentation

2.2 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 $h(x, y)$ in x and y separately in main function, $h(x, y)$ is separated into $h_1(x)$ and $h_2(y)$. This function is used to compute the GEF in x, i.e. $h_1(x)$. 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. $h_2(y)$ 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 ($m(x, y)$ or $m'(x, y)$), the original image, sigma, and threshold. The output is the visualized segmentation result. This function does not call any other functions.

3. Algorithms

We implemented Gabor filter following by a smoothing filter for the purpose of texture segmentation.

3.1 Garbor Filter:

We first apply Garbor filter to the input image:

$$m(x,y) = [I(x,y) ** h(x,y)], \quad [1]$$

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

$$h(x,y) = g(x, y) \exp[j2\pi F(x\cos\theta + y\sin\theta)] = g(x, y) \exp[j2\pi (Ux + Vy)]. \quad [2]$$

In equation 2, θ is the orientation of sinusoid, and g is a circularly-symmetric Gaussian:

$$g(x,y) = \frac{1}{2\pi\sigma^2} \exp\{\frac{-(x^2+y^2)}{2\sigma^2}\} \quad [3]$$

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

$$h(x,y) = h_1(x)\, h_2(y) \quad [4]$$

$$h_1(x) = \frac{1}{2\pi\sigma^2} \exp\{\frac{-x^2}{2\sigma^2}\} \exp\{j2\pi F\, x\cos\theta\} \quad [5]$$

$$h_2(y) = \frac{1}{2\pi\sigma^2} \exp\{\frac{-y^2}{2\sigma^2}\} \exp\{j2\pi F\, y\, \sin\theta\} \quad [6]$$

Then the Gabor filter can be applied to the input image through three steps:

- $i_1(x,y) = i(x,y) * h_1(x)$

$$= \sum_{x'=-2\sigma}^{+2\sigma} i(x - x', y)\ \widehat{h_1}(x') \quad [7]$$

- $i_2(x,y) = i_1(x,y) * h_2(y)$

$$= \sum_{y'=-2\sigma}^{+2\sigma} i_1(x, y - y')\ \widehat{h_2}(y') \quad [8]$$

- $m(x,y) = |\, i_2(x,y)\,|$ [9]

Thus the Gabor filter has a width of $(4\sigma + 1)$.

3.2 Smoothing Filter

Smoothing filter will be applied to m(x,y) when m(x,y) is noisy for better segmentation.

$$m'(x,y) = m(x,y) * g'(x,y) \quad [10]$$

where g′ is another circular-symmetric Gaussian defined in equation 3 using a different σ.

3.3 Segmentation

Based on the result of the step function m(x,y) or m'(x,y), we selected a threshold that can separate the step function into two parts. We then overlay it with the original image for segmentation visualization.

3.4 Parameters

Parameters used for each image are shown in table 1.

	Gabor Filter			Smooth Filter	Segmentation
	F	θ	σ	σ	Threshold
Texture 1	0.042	0	24	24	3.6e-9
Texture 2	0.059	135	8	24	9.1e-8
d9d77	0.063	60	36	/	0.5e-3
d4d29	0.6038	-50.5	8	30	9.9e-9

Table 1 Parameters to segment different images used in this project.

4. Results

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

4.1 Image 'texture1':

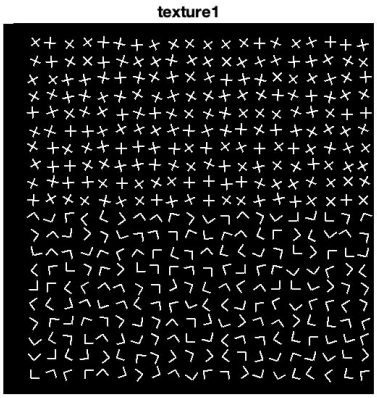

Figure 2. binary image of 'texture1'.

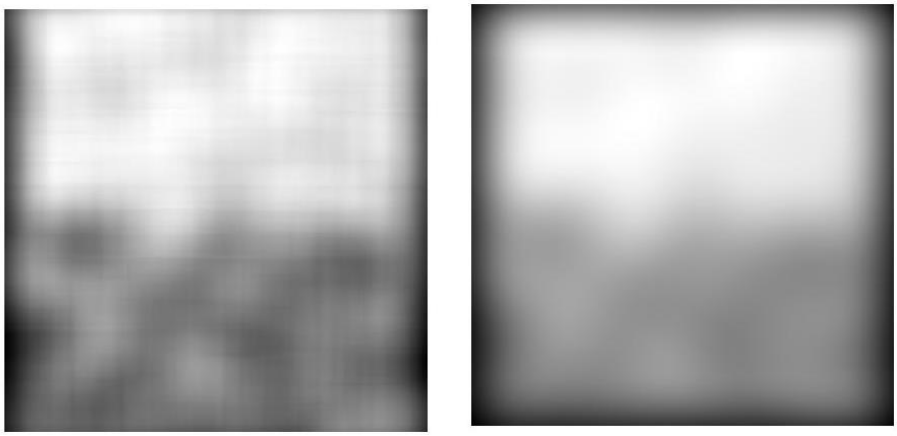

Figure 3. grayscale image of Gabor filter (left) and smoothing filter (right).

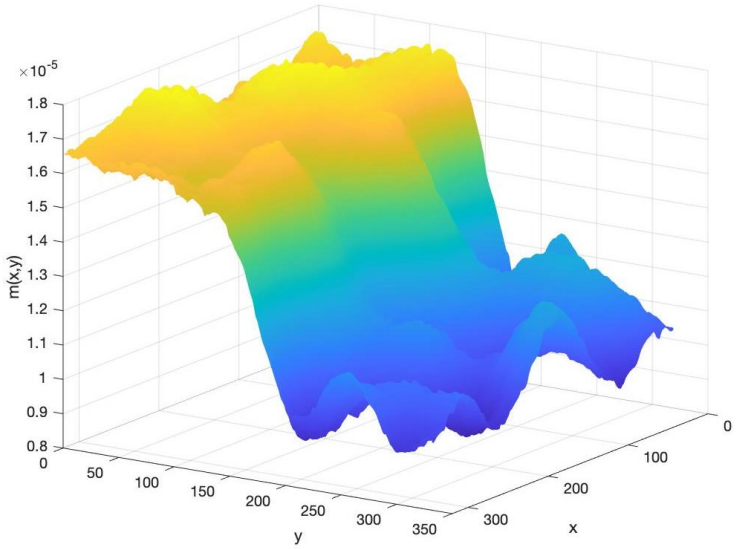

Figure 4. 3D plot of 'texture 1' after Gabor filter.

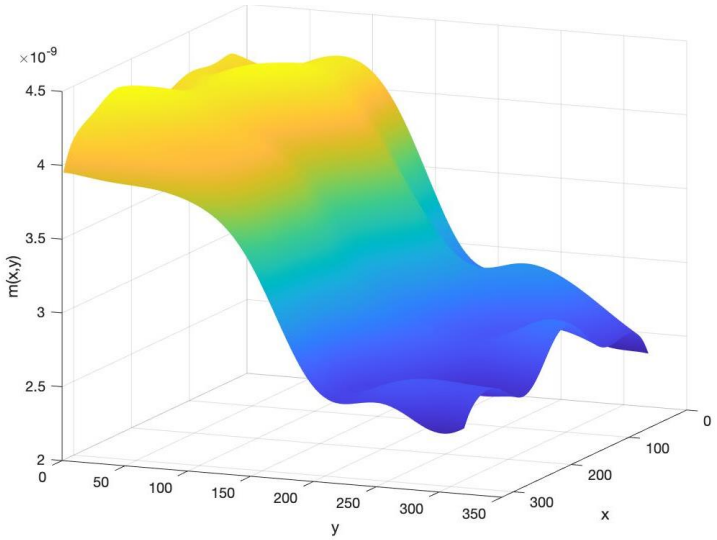

Figure 5. 3D plot of 'texture 1' after Gabor filter and smoothing filter.

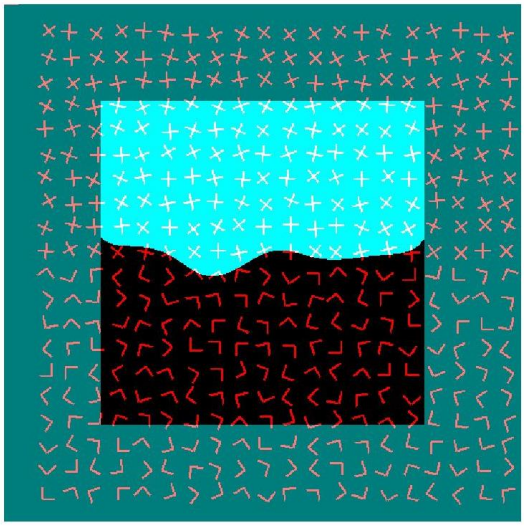

Figure 6. superimposition of texture segmentation of 'texture1'.

Observation: from Figure 3 (left), we can see that after processing Gabor filter, the grayscale image has shown two regions of different gray levels, but the boundary is somewhat blurring. After smoothing filter, the boundary becomes clearer and more straightforward as shown in Figure 3 (right). The 3D plot after Gabor filter illustrated in Figure 4 shows a sharp step, which implies a good texture segmentation of Gabor filter application with used parameters. Similar behavior can be found in Figure 5, the 3D plot after smoothing filter turns into a smoother slope. The final segmentation image shows a slight curve as the boundary due to the randomness of the texture's directions at the boundary.

4.2 image 'texture2':

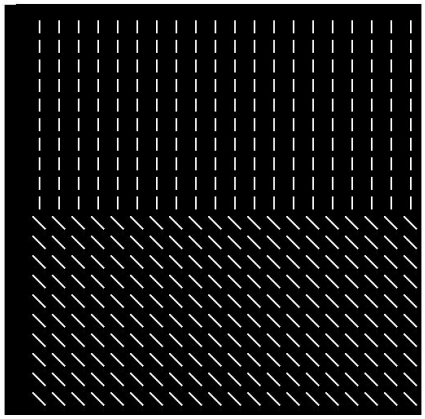

Figure 7. binary image of 'texture 2'.

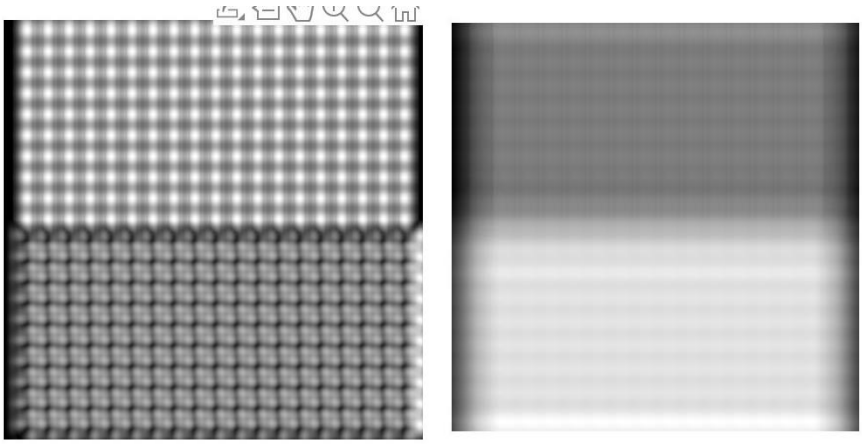

Figure 8. grayscale image of Gabor filter (left) and smoothing filter (right).

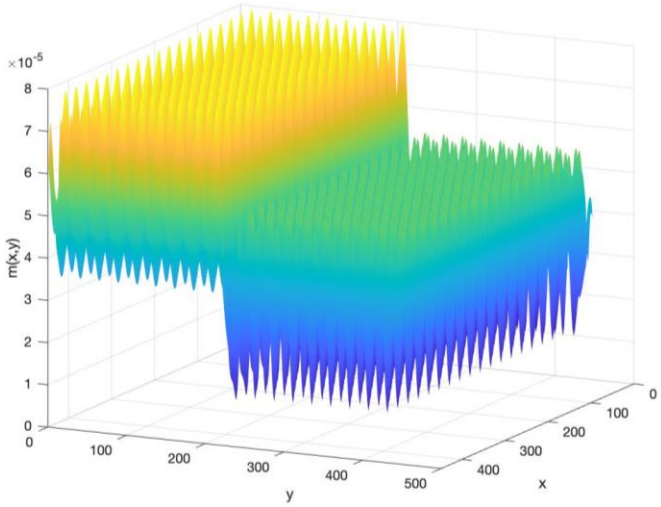

Figure 9. 3D plot of 'texture 2' after Gabor filter.

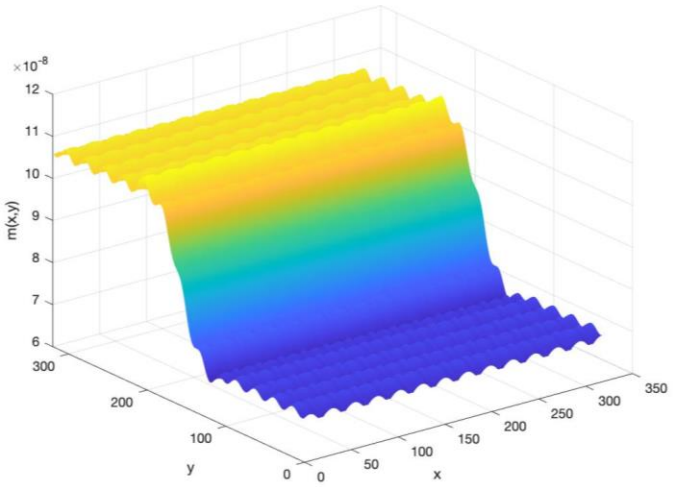

Figure 10. 3D plot of 'texture 2' after Gabor filter and smoothing filter.

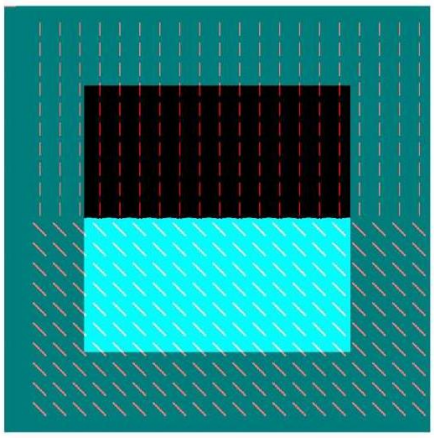

Figure 11. superimposition of texture segmentation of 'texture2'.

Observation: from Figure 8 (left), we can see that after processing Gabor filter, the grayscale image has shown two regions of different gray levels, but the boundary is somewhat blurring. With further smoothing filter, the boundary becomes clearer and more straightforward as shown in Figure 8 (right). The 3D plot after Gabor filter illustrated in Figure 9 looks like two blocks with sharp needle-like peaks, and there is a sharp step between two parts of blocks. Similar behavior can be found in Figure 10, the 3D plot after smoothing filter turns into a smoother slope. The final segmentation image shows better boundary approximately as a straight line, since the uniform and regularity of the texture direction.

4.3 image ‘d9d77’:

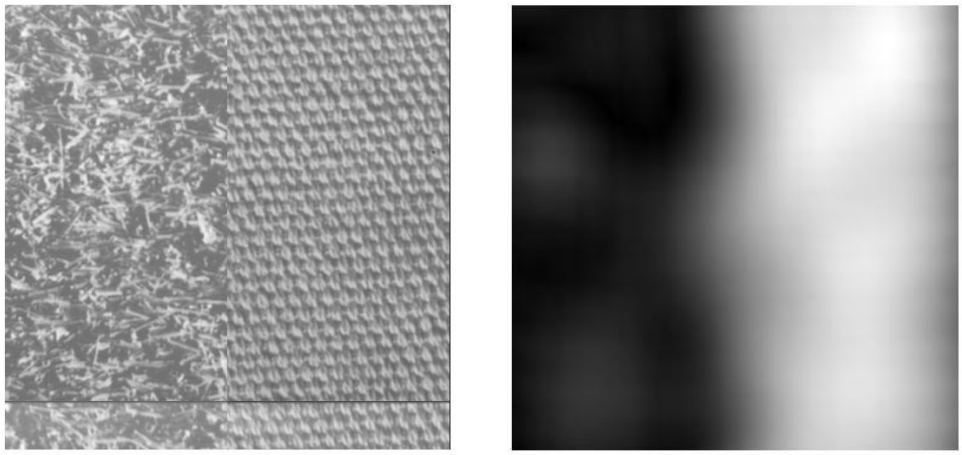

Figure 12. grayscale image of ‘d9d77’ (left) and after Gabor filter (right).

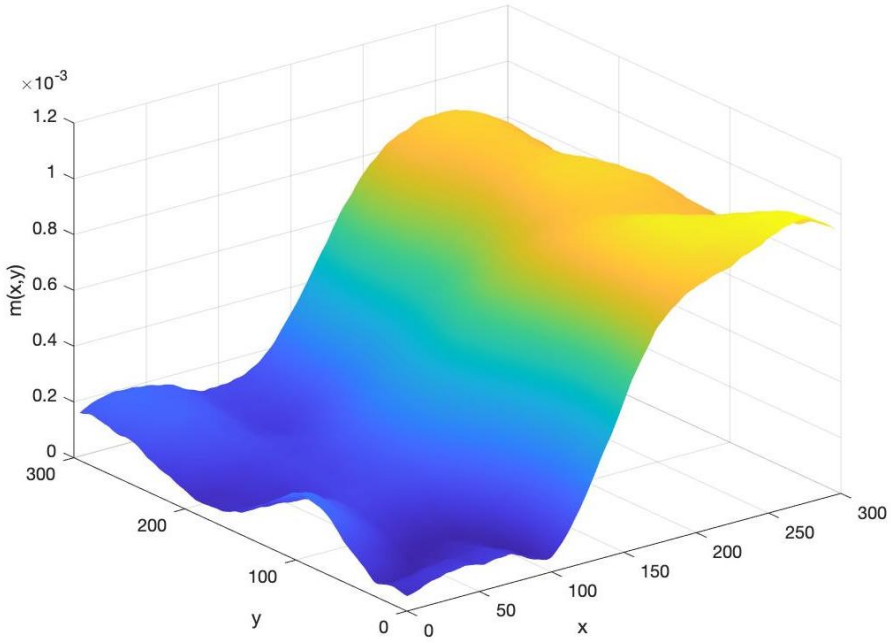

Figure 13. 3D plot of ‘d9d77’ after Gabor filter.

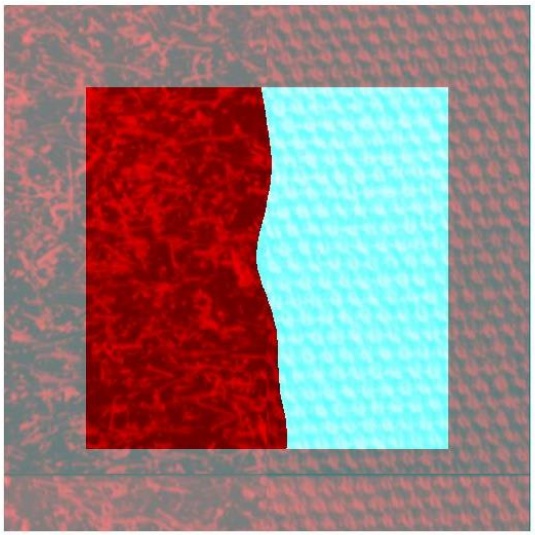

Figure 14. superimposition of texture segmentation of 'd9d77'.

Observation: from Figure 12 (right), we can see that after processing Gabor filter, the grayscale image has shown two regions of different gray levels, but the boundary is somewhat blurring. We assume that this is because of the randomness of texture on the left, however, texture on the right is very uniformly ordered. The 3D plot after Gabor filter illustrated in Figure 13 shows a very smoother slope, which implies a best set of parameters we have chosen so far. The final segmentation image shows a curve-like boundary due to the randomness of texture on the left.

4.4 image 'd4d29':

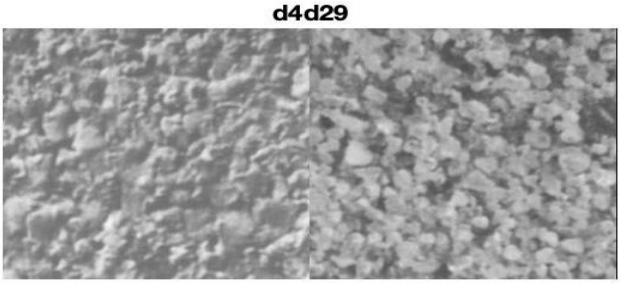

Figure 15. binary image of 'd4d29'.

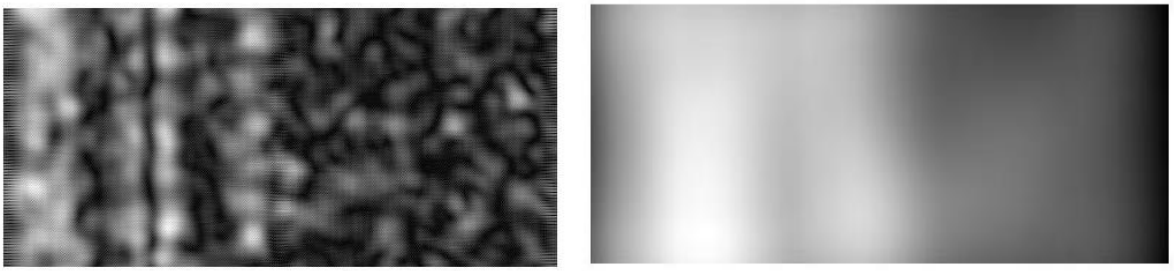

Figure 16. grayscale image of Gabor filter (left) and smoothing filter (right) of 'd4d29'.

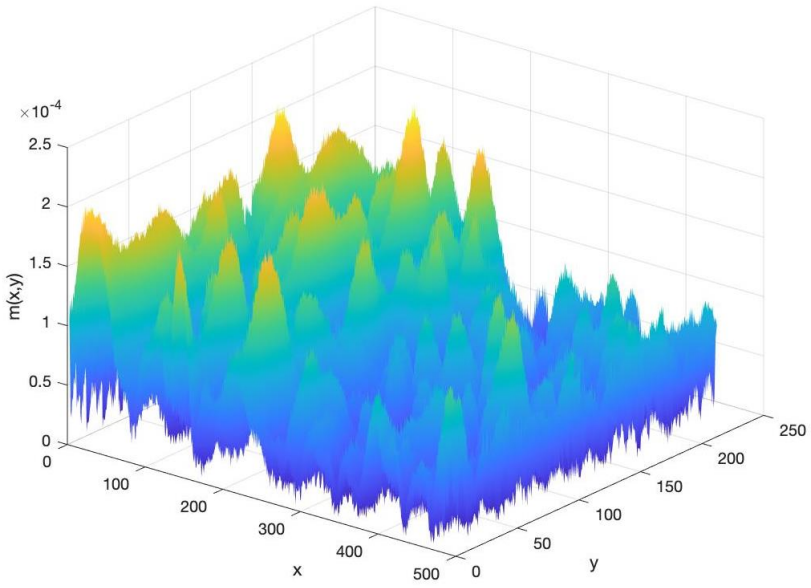

Figure 17. 3D plot of 'd4d29' after Gabor filter.

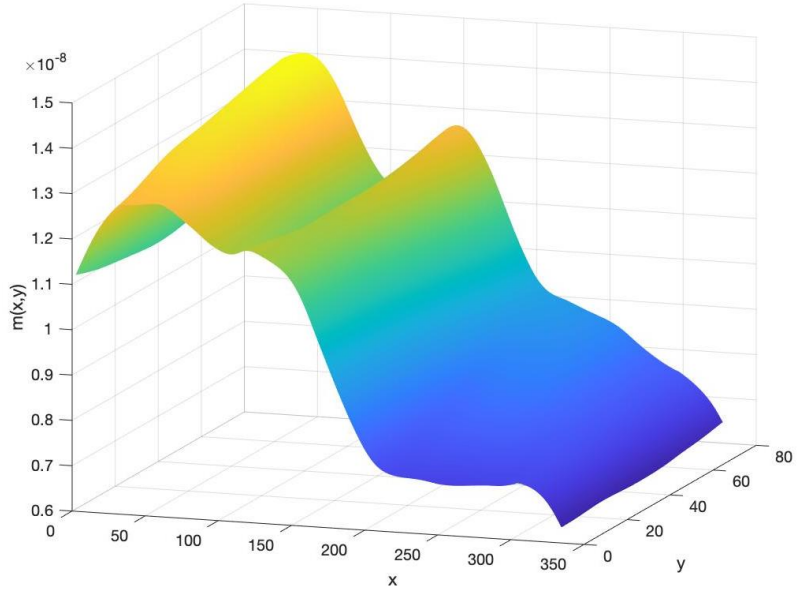

Figure 18. 3D plot of 'd4d29' after Gabor filter and smoothing filter.

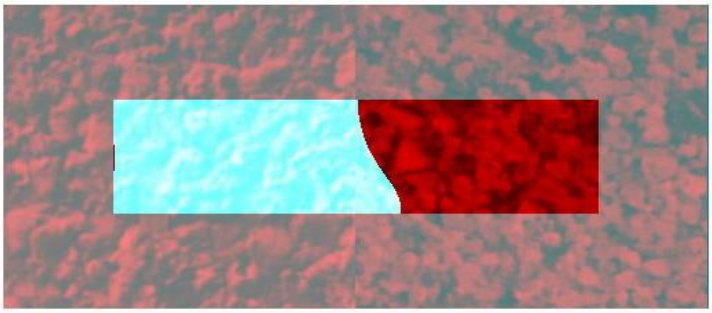

Figure 20. superimposition of texture segmentation of 'd4d29'.

Observation: the last image is the hardest one to process in our project. From Figure 16 (left) , we can see that after processing Gabor filter, the grayscale image shows less discriminative information as our previous tasks. With further smoothing, Figure 16 (right) reveals clearer discrimination of two regions with different gray levels, but the boundary is somewhat blurring. The 3D plot after Gabor filter illustrated in Figure 17 shows forest-like regions with two altitudes, and there is a sharp step between two forests. In Figure 18, the 3D plot after smoothing filter turns into a smoother slope but with small hills. The final segmentation image gives a curving boundary due to the high similarity of textures.

4.5 Discussion

In this project, the second image whose texture includes '-' and '\' was best separated, since they are well aligned. The edge separating the two textures '+' and 'L' in the first image was slightly curved due to their random directions. This also happened in image 'd9d77' because the two textures in 'd9d77' are closer to each other. Separating textures in the last image 'd4d29' was the hardest because the two textures are very similar.

Since the edge ($2 * \sigma$) could not be properly processed at all, values in these regions were set to 0. As a result, pixels located up to $4 * \sigma$ to the edge could all be affected as up to half of the values in equation 7 and 8 were zeros. Such region could be large when σ values are relatively big. The x and y limits delineating the segmentation for each image were given in table 2.

	x_1	x_2	y_1	y_2
Texture 1	97	416	97	416
Texture 2	97	416	97	416
d9d77	81	432	81	433
d4d29	81	176	41	433

Table 2. x and y limits delineating the segmentation for each image.

5. Conclusion

In conclusion, the project implemented single Gabor filters as well as smoothing filter on different bipartite textured images. Our experiments have validated that Gabor filter has selectivity for orientation, F and the standard deviation of Gaussian part of GEF. When parameters are selected properly, Gabor filter will output a step function where the two levels correspond to the two textures. A smooth filter will be needed when the output of Gabor filter is noisy. With a clean step function, we can select a threshold in the middle of the step to split the two textures. Since the edge could not be processed by Gabor filter or smooth filter, the segmentation could only happen in the center.