CMPEN 555: Digital Image Processing II

Computer Project # 4:

Texture Segmentation

Naichao Yin, Wenrui Wang, Zekai Liu

Date:4/10/2020

A. Objectives

1. Learn how to use MATLAB for general image processing.

2. Apply the Gabor filter and the smoothing filter to the images

3. Discriminate one texture from anther after segment the filtered image

B. Methods

1) Gabor Filter

We already know that the Fourier transform is a powerful operation in signal and image processing, which can help us transform images from the spatial domain to the frequency domain and extract features from the spatial domain that are difficult to detect. However, after Fourier transform, the frequency information of images in different positions are often mixed, but Gabor filter can improve that, which is an effective texture segmentation tool.

The main idea of Gabor filter is that different texture generally has its individual center frequency and bandwidth. According to these properties we can design a set of Gabor filter for specific texture segmentation, each filter only allow one frequency pass, and make the other restrained. Gabor filter is a bandpass filter, whose impulse response function (Gabor function) is the product of complex exponential function with Gaussian function. In our project, we use a Sinusoidal function superimposed with a Circularly-Symmetric Gaussian function, as shown below in Equation 1.1 and 1.2.

$$h(x,y)=g(x,y)\cdot\exp[j2\pi F(x\cos\theta+y\sin\theta)]=g(x,y)\cdot\exp[j2\pi(Ux+Vy)] \qquad (1.1)$$

$$g(x,y) = \frac{1}{2\pi\sigma^2} \cdot \exp\left\{ \frac{-\left(x^2+y^2\right)}{2\sigma^2} \right\} \tag{1.2}$$

Where **h** is a Gabor Elementary Function (GEF), and **g** is a circularly-symmetric Gaussian function using a different standard deviation **σ**, which control the shape of Gaussian. We create this Gaussian function in our MATLAB function **"circleGaussian(sigma)"** to compute discrete values for a 2D symmetric circular gaussian distribution with standard deviation. Parameters **θ** and **F** in **h** specifies the texture direction of the Gabor function and determine the center frequency information. We generate a 2D discrete GEF with specific width in our MATLAB function **"gefGenerator(F, theta, sigma)"**.

$$m(x,y) = | I(x,y) * h(x,y) | \tag{1.3}$$

To apply the Gabor filter appropriately, we truncated Gaussian function modulated by sinusoid over a [-2σ, +2σ] range, so it has width 4σ+1 pixels in either direction in the GEF. After this, we do straightforward spatial convolution between input image **I** and GEF **h**, and apply magnitude operator |·| to this output in our MATLAB function **"gaborFilter(Image,F,theta,sigma)"** to perform Gabor filtering on the input image.

$$h(x,y) = h_1(x,y) \cdot h_2(x,y) \tag{1.4}$$

$$I_1(x,y) = \sum_{x'=-2\sigma}^{+2\sigma} I(x-x',y) \cdot h_1(x') \tag{1.5}$$

$$I_2(x,y) = \sum_{y'=-2\sigma}^{+2\sigma} I_1(x,y-y') \cdot h_2(y') \tag{1.6}$$

$$m(x,y) = \left| I_2(x,y) \right| \tag{1.7}$$

To convolve efficiently, we use steps below (Equation 1.4-1.7) in our MATLAB function **"convolve2D(Image,Kernel)"**, where input **I** as "Image" parameter and **h** as "Kernel", to separate the GEF **h** from **h(x)** and **h(y)**, and convolve it in one dimension each time instead of two dimensions.

2) **Smoothing Filter**

Image smoothing, also known as "blur", is to suppress the noise of the target image while preserving the details of the image. The purpose of smoothing filter is to eliminate the noise of the image and extract the features of the target region. The requirements of the filter are as follows: the important feature of the image (such as contour and edge) cannot be damaged, and the filtered image needs to be clearer.

$$g^{'}(x,y)=\frac{1}{2\pi\sigma^{2}}\cdot\exp\left\{\frac{-\left(x^{2}+y^{2}\right)}{2\sigma^{2}}\right\} \tag{2.1}$$

In our project, we use the Gaussian filter as our smoothing filter (Equation 2.1). The principle of the Gaussian filter is to take the average of the pixels in the filter mask as the output, and the pixels closer to the center have higher weights. The smoothness of the image after Gaussian filtering depends on the standard deviation **σ**.

$$m^{'}(x,y)=m(x,y)*g^{'}(x,y) \tag{2.2}$$

To apply the Smoothing filter appropriately, we still truncated Gaussian function modulated by sinusoid over a [-2σ, +2σ] range like the previous functions. After this, we do straightforward spatial convolution between Gabor filter **m** and Smoothing filter **g’** (Equation 2.2)in our MATLAB function **“gaussianSmooth(Image,var)”** to perform smoothing on the target image.

3) Texture Segmentation

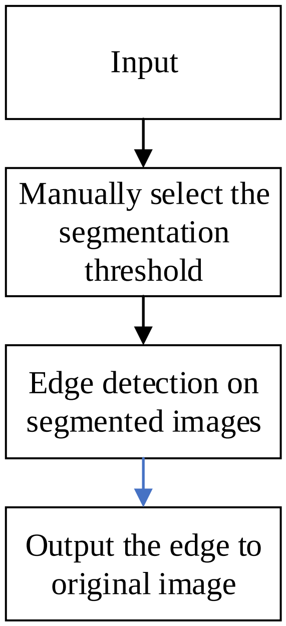
Input
Manually select the segmentation threshold
Edge detection on segmented images
Output the edge to original image

Figure 1.1. Flowchart of Texture Segmentation

Threshold the filtered output **m’** to give a segmented image, where the goal is to discriminate one texture from another. In our MATLAB function **“segment(GEF, Image, The)”**, we first create the new image **img_s**, which is the same size as input image **m’**, then set the threshold to separate the target pixels (the gray-scale value is equal to or greater than the threshold) from GEF and input them in the image **img_s**.

Since different textures in a smoothed image have individual gray-scale, we can easily divide them into different regions by applying a Canny edge detector with threshold **0.5** which can detect edges with noise suppressed at the same time and always using after Gaussian filtering. Finally, we superimpose the segmented result onto the original image to obtain the final output after texture segmentation.

4) Display Result

When displaying results, we first zero out the unprocessed perimeter areas. For the display of the complex gray-scale images **m** and **m’** after Gabor filtering, we using our MATLAB function **“displayGabor(ComplexImg)”** to normalize these images, grab the real part of the filtered image and scale the gray-scale value from 0 to 255. Also, we use the function **“mesh”** to generate the 3D mesh surface of image **m** and **m’**, which is specified by each pixel and its gray-scale value.

5) MATLAB and Parameters

There are 9 MATLAB files in this project: **“p4Main.m”**, **“Q3Q4.m”**, **“circleGaussian.m”**, **“gefGenerator.m”**, **“convolve2D.m”**, **“displayGabor.m”**, **“gaborFilter.m”**, **“gaussianSmooth.m”**, and **“segment.m”**. The main codes are in the file **“p4Main.m”** and the file **“Q3Q4.m”**, which include the whole texture segmentation process of test 1,2 and test 3,4.

In each test, we call the file **“gaborFilter.m”**, **“gaussianSmooth.m”**, **“segment.m”** and **“displayGabor.m”** in sequence to do Gabor filtering, Smooth filtering, texture segmentation and display our final result.

To make full use of the characteristics of the Gabor filter, Smoothing filter and Segmentation, it is necessary to design specific parameters manually. Gabor filter with different texture orientation and spacing to extract features, Smoothing filter with different standard variances to achieve the different smoothing effects, and Segmentation with different thresholds to discriminate one texture from another. The specific parameters used in each task are shown in the following Table 1.

	F(cycles/pixel)	**Θ(°)**	**σ(Gabor)**	**σ(Smooth)**	**Threshold**
Test 1 (|, \\)	0.059	135	8	24	0.04
Test 2 (+, L)	0.042	0	24	24	0.045
Test 3 (d9, d77)	0.063	60	36	36(*)	0.08
Test 4 (d4, d29)	0.6038	- 50.5	8	40	0.024

Table 1. Parameters of 4 Tests

C. Result

1) Tests one

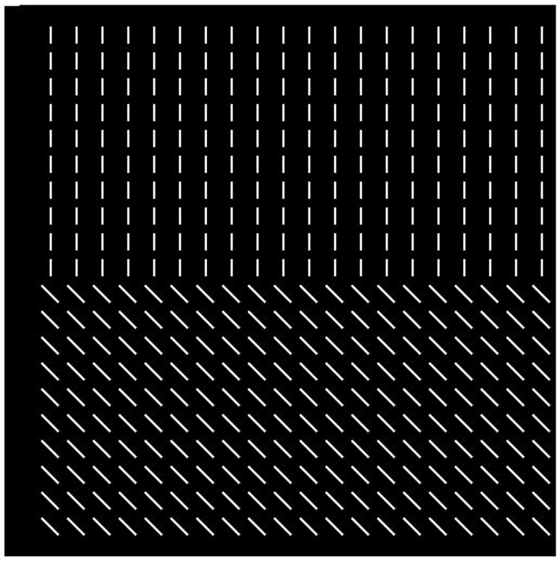

Figure 2.1.1. Original Image of Texture1

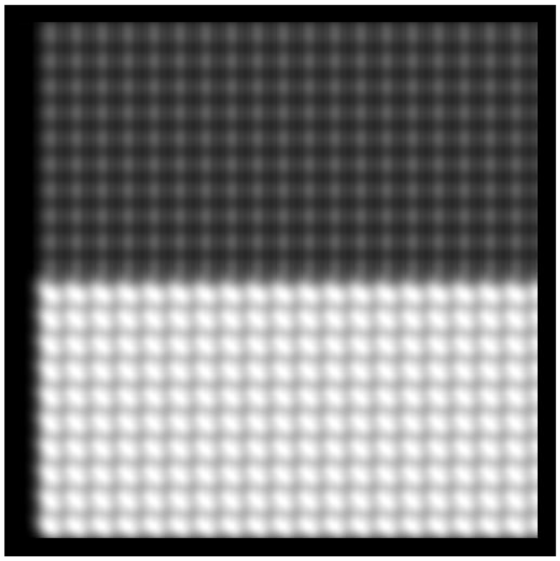

Figure 2.1.2 Texture1 After Gabor filter

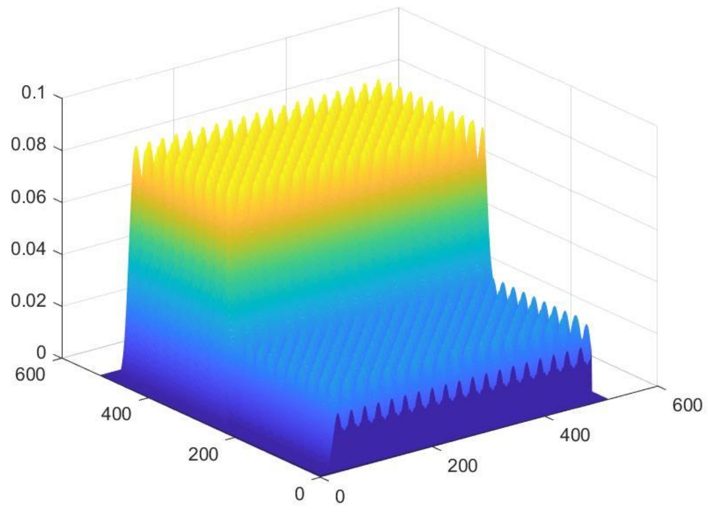

Figure 2.1.3 3D plot of Gabor filtered Image

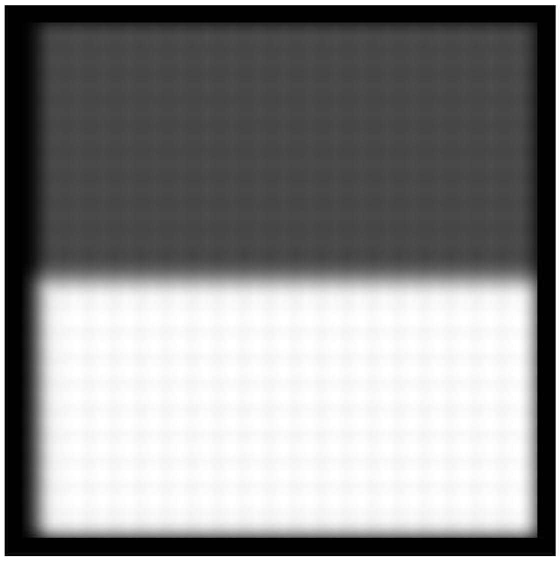

Figure 2.1.4 Texture1 After Smoothing filter

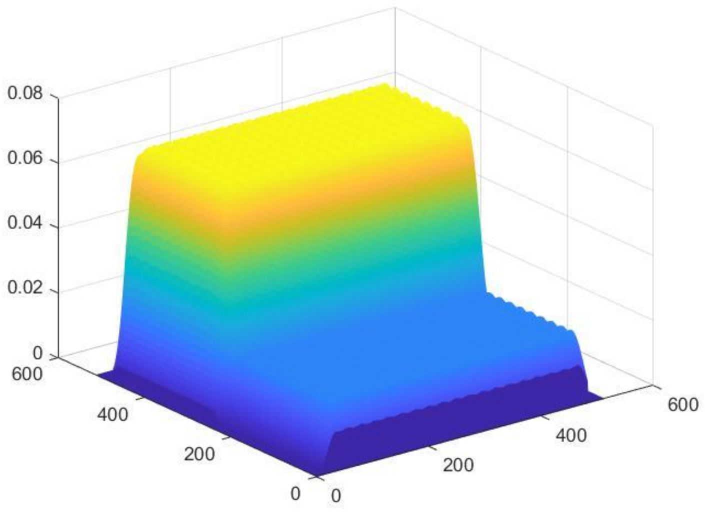

Figure 2.1.5 3D plot of Smoothing filtered Image

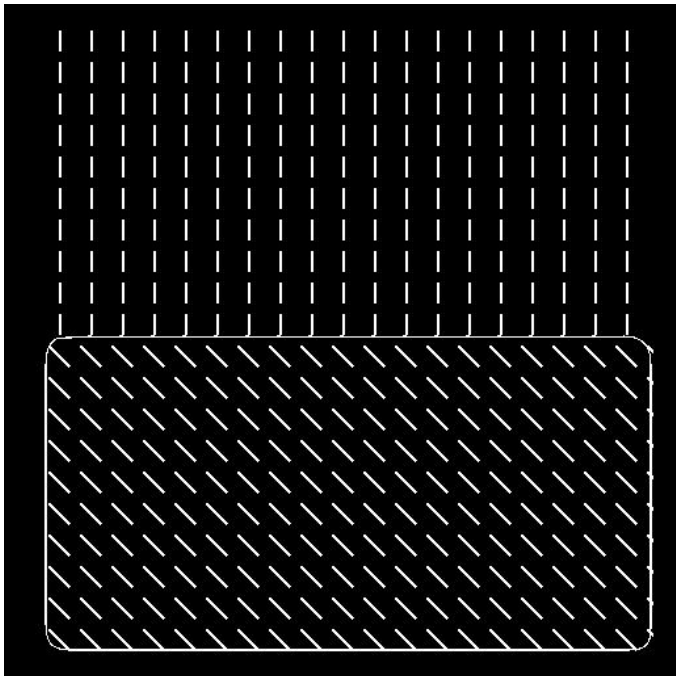

Figure 2.1.6 Original Image with Segmentation

By comparing the original image (Figure 2.1.1) with the segmented image (Figure 2.1.6), it can be observed that the segmentation acts on the whole image. In the gray-scale and 3D plot of Gabor filtered image (Figure 2.1.2 and 2.1.3), we can see that the Gabor filter only allows the “|” texture to pass through, while the "\" texture is completely inhibited. Although the segment connects to the **"|"** at the top and **"\"** at the bottom, we can still conclude that the algorithm and its parameters do a good job of segmenting the **"|"** and **"\"** textures in the original image. The limitation of x and y is x in[17, 495] and y in [17, 495].

2) Tests two

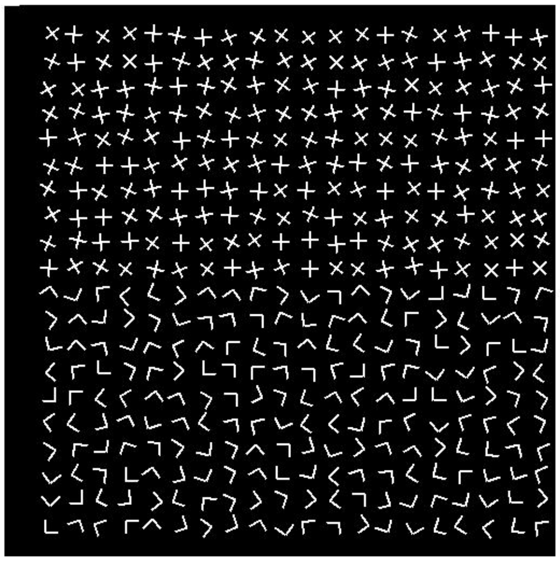

Figure 2.2.1 Original Image of Texture2

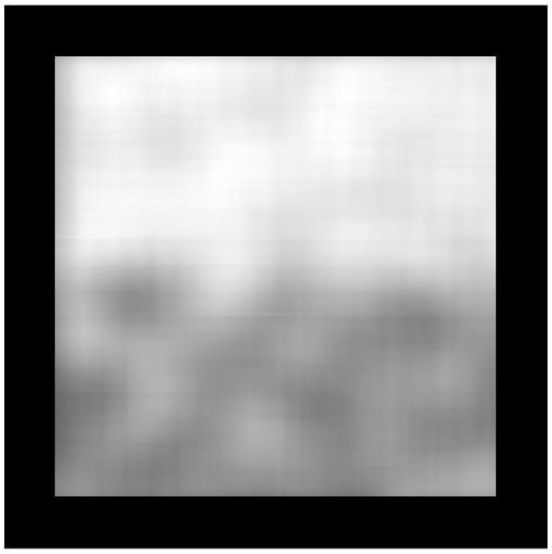

Figure 2.2.2 Texture2 After Gabor filter

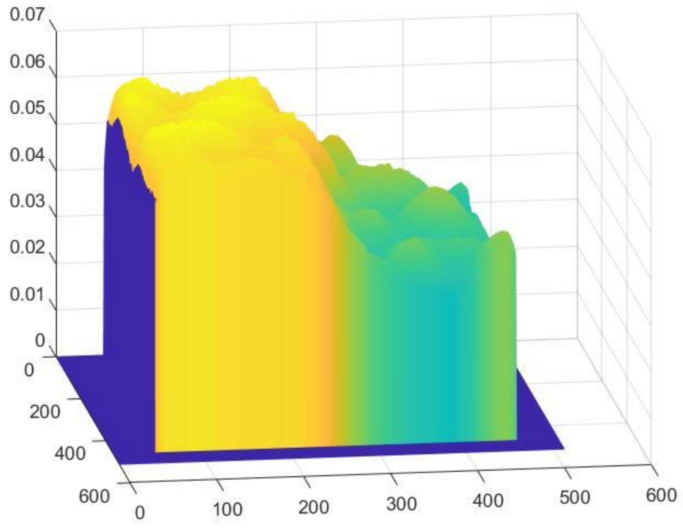

Figure 2.2.3 3D plot of Gabor filtered Image

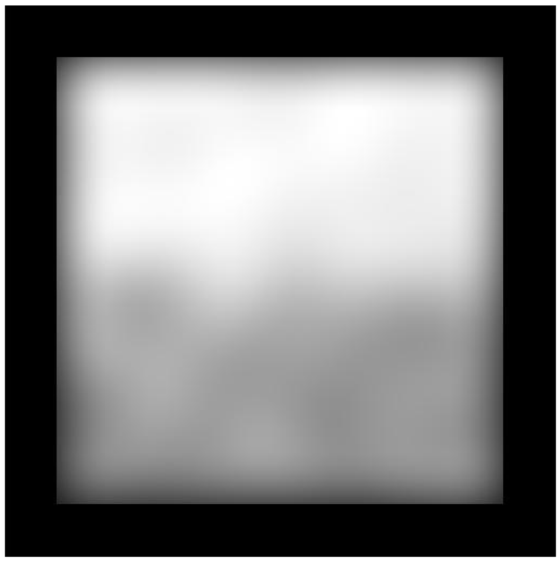

Figure 2.2.4 Texture2 After Smoothing filter

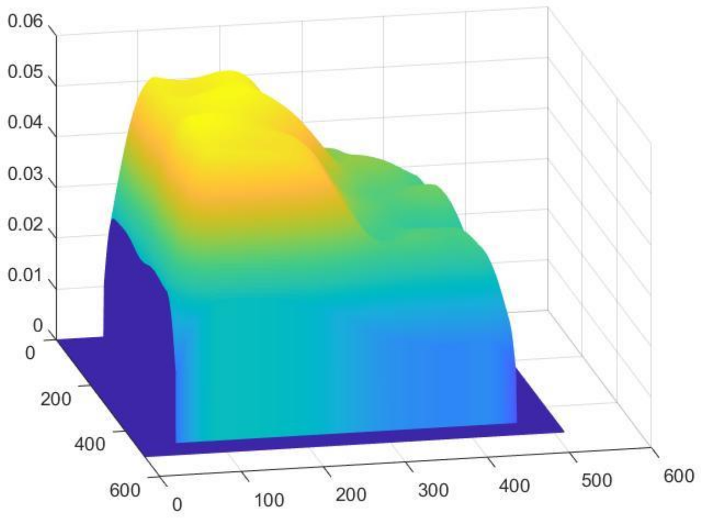

Figure 2.2.5 3D plot of Smoothing filtered Image

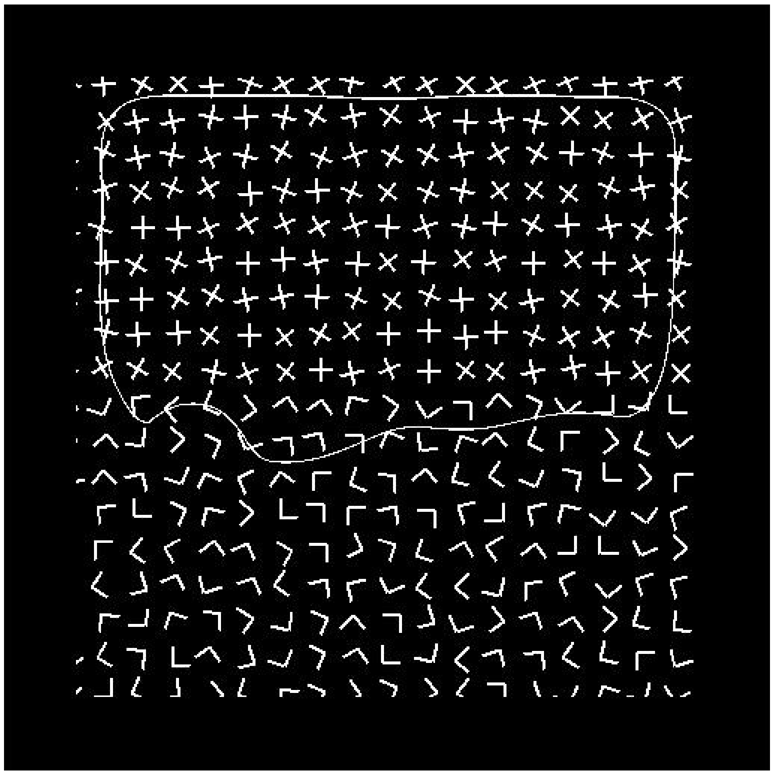

Figure 2.2.6 Original Image with Segmentation

By comparing the original image (Figure 2.2.1) with the segmented image (Figure 2.2.6), it can be seen that the segmentation only affects the center part of the image. In the gray-scale and 3D plot of Gabor filtered image (Figure 2.2.2 and 2.2.3), we can see that the Gabor filter allows the **“+”** texture to pass through, but the **“L”** texture is not greatly suppressed. The rounded rectangle box in Figure 2.2.4 does not include the whole region of **“+”**, and it also contains a portion of the “L” region. However, we can still conclude that the algorithm and its parameters do not bad of segmenting the **“+”** and **“L”** textures in the original image. The limitation of x and y is x in [49, 463] and y in [49, 463].

3) Tests three

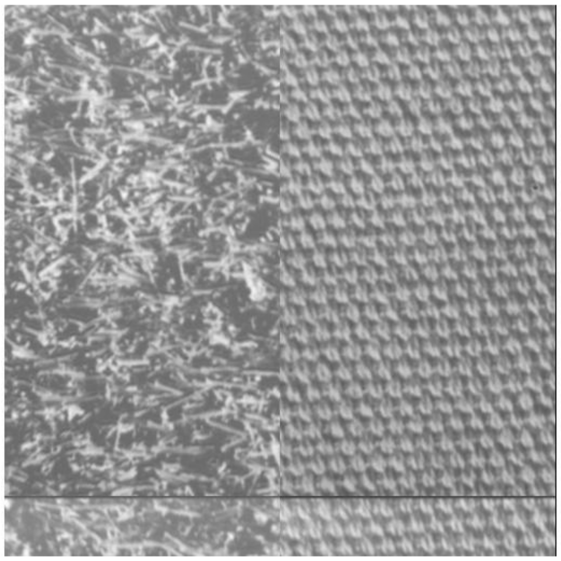

Figure 2.3.1 Original Image of “d9d77”

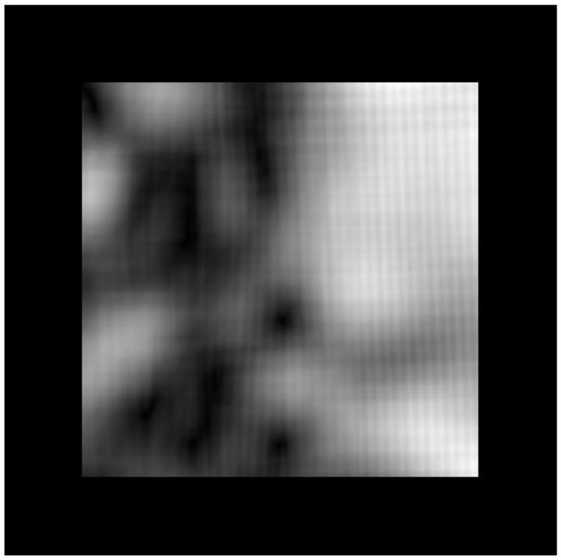

Figure 2.3.2 “d9d77” After Gabor filter

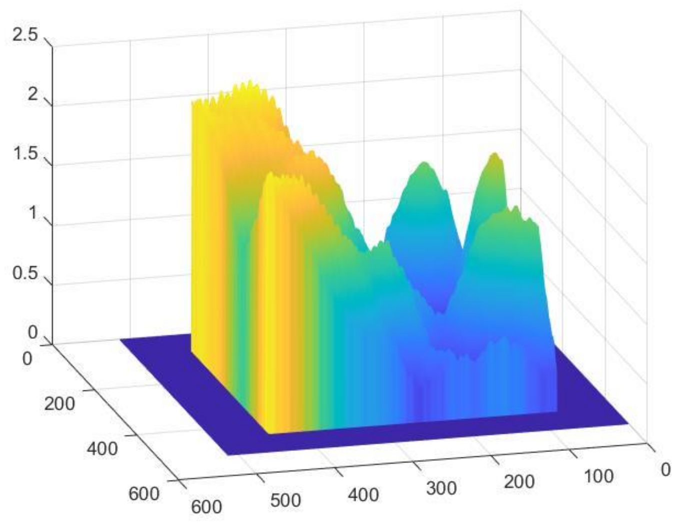

Figure 2.3.3 3D plot of Gabor filtered Image

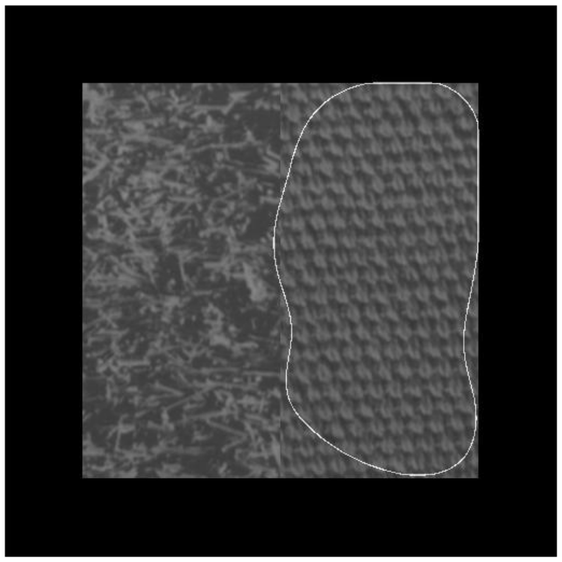

Figure 2.3.4 Original Image with Segmentation

By comparing the original image (Figure 2.3.1) with the segmented image (Figure 2.3.4), it can be observed that the segmentation only affects the center part of the image and the region on the outer perimeter of the image that cannot be processed gets larger. In the gray-scale and 3D plot of

Gabor filtered image (Figure 2.3.2 and 2.3.3), we can see that the Gabor filter allows the **"d9"(grass lawn)** texture to pass through, but the more it moves toward the center, the more inhibited it is, even worse than the **"d77"(cotton canvas)** texture on the left. The rounded rectangle box in Figure 2.3.3 does not include the whole region of **"d77"**, but we can still get segmentation. The limitation of x and y is x in [73, 439] and y in [73, 440] (the image is 512*513).

4) Tests four

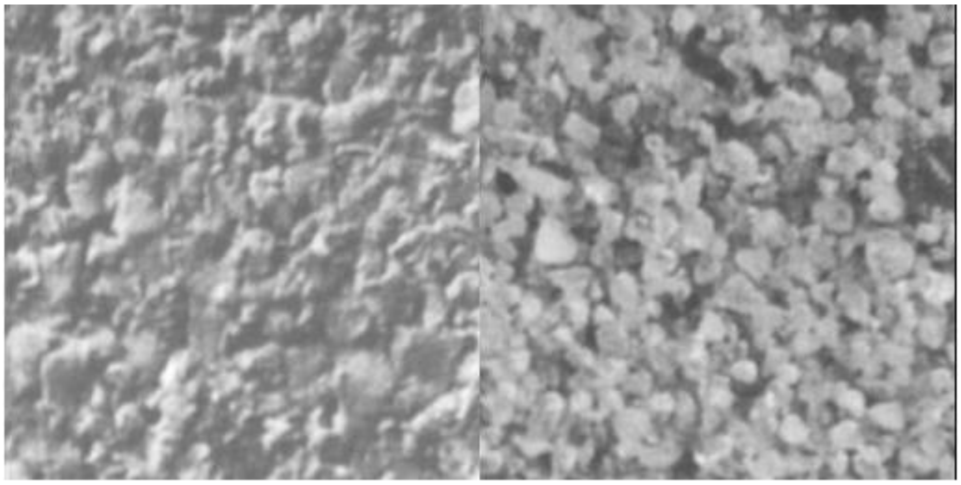

Figure 2.4.1 Original Image of "d4d29"

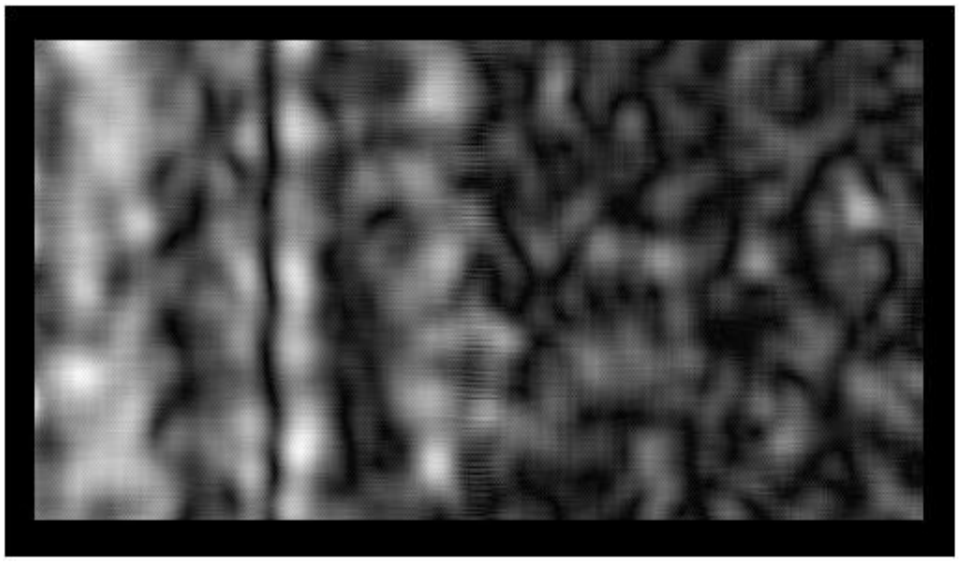

Figure 2.4.2 "d4d29" After Gabor filter

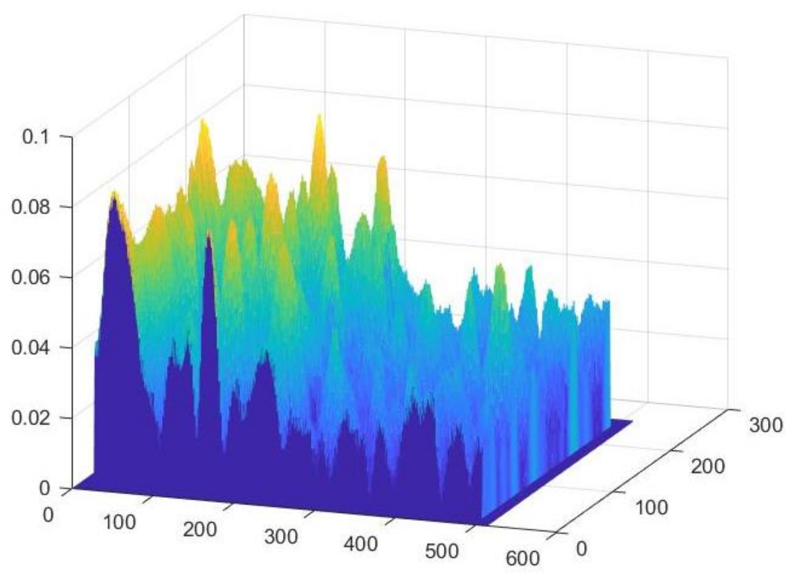

Figure 2.4.3 3D plot of Gabor filtered Image

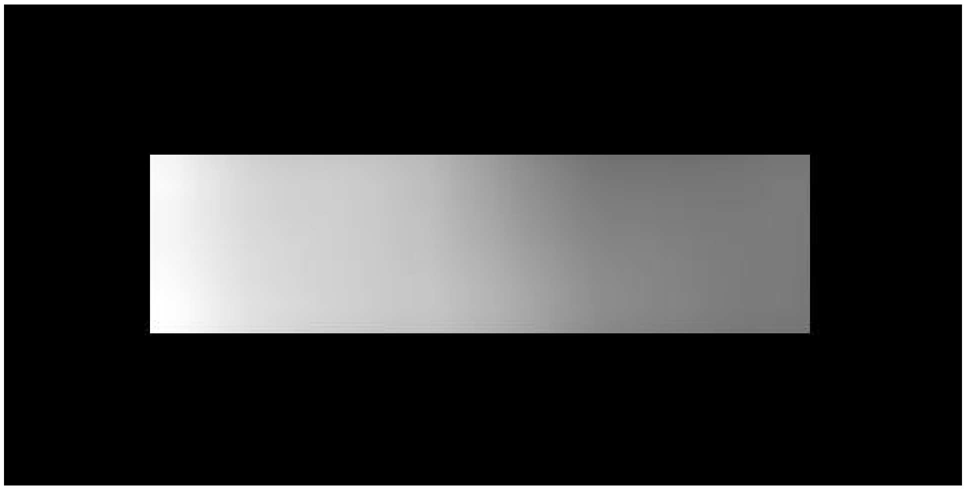

Figure 2.4.4 “d4d29” After Smoothing filter

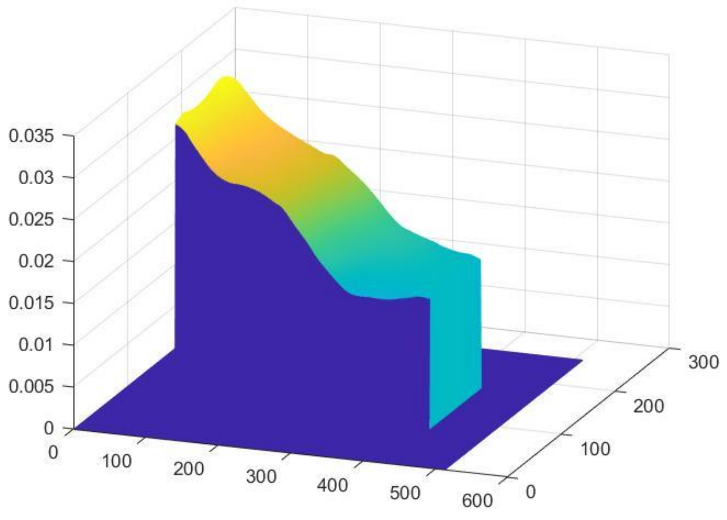

Figure 2.4.5 3D plot of Smoothing filtered Image

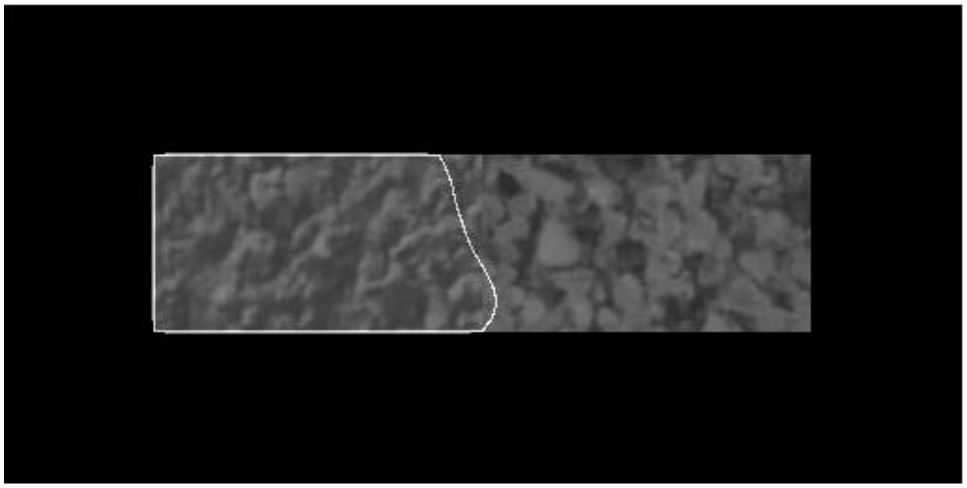

Figure 2.4.6 Original Image with Segmentation

By comparing the original image (Figure 2.4.1) with the segmented image (Figure 2.4.6), it can be seen that the segmentation only affects a small central part of the image. In the gray-scale and 3D plot of Gabor filtered image, we can see that the Gabor filter allows the **"d4"(pressed cork)** texture to pass through, while the **"d29"(beach sand)** texture is inhibited. However, the image after the Gabor filter has severe noise (Figure 2.4.2 and 2.4.3), which needs to be smoothed by postfilter (Gaussian filter). The rounded rectangle box in Figure 2.4.6 fits well in the region of **“d4”** and successfully pull textures apart after the smoothed image (Figure 2.4.4). The limitation of x and y is x in [81, 175] and y in [81, 432].

D. Conclusion

Gabor is a linear filter used for edge extraction. Its frequency and direction expressions are similar to the human visual system. It can provide good direction selection and scale selection characteristics. Gabor wavelet is sensitive to the edge of the image, can provide good direction selection and scale selection characteristics, and is not sensitive to changes in illumination, can provide good adaptability to changes in illumination. Therefore, it is very suitable for texture analysis. Gabor transform is a short-time Fourier transform method, its essence is to add a window function in the Fourier transform, through the window function to achieve the time-frequency analysis of the signal. When the Gaussian function is selected as the window function, the short-time Fourier transform is called Gabor transform. The result of the Gabor filter maybe not optical for segmenting sometimes. We can use Gaussian Smooth to deal with it, which can reduce image noise and reduce the level of detail.