CSE585/EE555: Digital Image Processing II
Computer Project # 4
Texture Segmentation

Saptarashmi Bandyopadhyay ; Bharadwaj Ravichandran

04/13/2019

---

1 Objectives

The main objective of this project is to implement the Gabor Filter for texture segmentation followed by the Smoothing filter to distinguish the texture regions clearly.

2 Methods

2.1 Gabor Filter for Texture Segmentation

The Gabor filter is a bandpass filter that is useful for texture analysis of images. A Gaussian function, g(.) is multiplied with a complex sinusoidal expression to obtain a pre-computed Gabor Elementary Function (GEF), h(.) as described below. Then the 2-D convolution is implemented for every pixel in the image by using the GEF to obtain the filtered image.

The Mathematical representation for the GEF is as

follows:

$$h(x,y) = g(x',y') \exp\,[j2\pi(Ux+Vy)]$$

The Gaussian function and the sinusoidal expression in the above equation have been defined below.

(a) Gaussian Function

The Gaussian function is centered at (0,0). It is defined as

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\Big\{-\frac{1}{2}\Big[\Big(\frac{x}{\sigma_x}\Big)^2 + \Big(\frac{y}{\sigma_y}\Big)^2\Big]\Big\}$$

$$(x',y') = (x\cos\phi \;+\; y\sin\phi\;,\; -x\sin\phi \;+\; y\cos\phi)$$

$\uparrow$

rotated spatial-domain coordinates

$$\begin{bmatrix}\cos\phi & \sin\phi \\ -\sin\phi & \cos\phi\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}\begin{bmatrix}x'\\y'\end{bmatrix}$$

$\uparrow$

rotation matrix

$\sigma_x\sigma_y \left(\lambda = \dfrac{\sigma_y}{\sigma_x}\right)$ control shape of Gaussian

If $\sigma_x = \sigma_y$, $g(.\ ,\ .)$ is symmetrical $\rightarrow \phi$ not needed

In our project, σ value has been provided, where $\sigma = \sigma_x = \sigma_y$. So ϕ value is not needed in the project.

(b) Sinusoidal part of GEF:

The sinusoidal part of GEF is a complex expression with real and imaginary components. It can be defined in the following way.

$$\exp\,[j2\pi(Ux+Vy)] \equiv \exp\,[j2\pi F(x\cos\theta + y\sin\theta)]$$

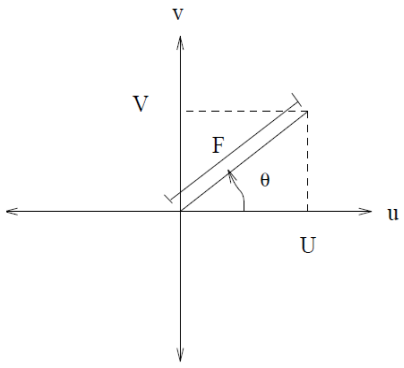

Figure 1: (U,V) Frequency Space

$$U = F\cos\theta \qquad V = F\sin\theta$$

$$(U, V) = \text{ particular 2-D Frequency}$$

$$\theta = \text{ orientation of sinusoid} = \tan^{-1}\left(\frac{V}{U}\right)$$

$$\phi = \text{ orientation of Gaussian}$$

The window size of the filter over each pixel in the image is $[-2\sigma, 2\sigma]$. The GEF h(.) is thus a pre-computed array of length 4σ +1 The Gabor filter for Texture Segmentation can thus be applied on image i(x,y) to generate the filtered image m(x,y) as follows

$$m(x, y) = |i(x, y) \ ** \ h(x, y)| \tag{1}$$

where h(x,y) is the GEF that can be expressed as

$$h(x, y) = g(x, y) * \exp\{j2\pi F(x\cos\theta + y\sin\theta)\} \tag{2}$$

Now,

$$h(x,y) = h(x)\ .\ h(y)\ ,\ \ g(x,y) = g(x)\ .\ g(y) \tag{3}$$

where

$$h(x) = g(x) * \exp\{j2\pi Fx\cos\theta\} \tag{4}$$

$$h(y) = g(y) * \exp\{j2\pi Fy\sin\theta\} \tag{5}$$

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma_x}\ \exp\Big\{-\frac{1}{2}\Big(\frac{x}{\sigma_x}\Big)^2\Big\} \tag{6}$$

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma_y}\ \exp\Big\{-\frac{1}{2}\Big(\frac{y}{\sigma_y}\Big)^2\Big\} \tag{7}$$

The 2-D convolution operation has been executed by first applying the 1-D convolution for each y in i(x,y) with the GEF to generate an intermediate $i_1(x,y)$. Again a 1-D convolution is applied for each x in $i_1(x,y)$ with the GEF to generate $i_2(x,y)$, the absolute value of which is the filtered image m(x,y).

The Gabor filter has been implemented in the MATLAB code GEF_3.m

The pseudocode of Gabor filter is as follows:

Objective: To perform on image i(x,y) with GEF h(x,y).

Input Parameters:
im: Binary Input Image in double format
theta: orientation of the GEF h(.)
sigma: standard deviation of the Gaussian function for Gabor filter
F: frequency value

Returned Result:
M_gaussian_scaled - Resulting image which is scaled after Gabor filtering.

Processing Flow:

1. Set two arrays full of zeros with the same dimension (M,N) as the input image to denote $i_1(x,y)$ and $i_2(x,y)$.

2. For each iteration i in the window of $4\sigma + 1$

a) Compute the Gaussian function g_1 where x is $i - 2\sigma - 1$

b) Compute the GEF h_1 as $h_1(i) = g_1(i) * e^{j2\pi F(i-2\sigma-1)cos(theta)}$

c) Compute the Gaussian function g_2 where y is $i - 2\sigma - 1$

d) Compute the GEF h_2 as $h_2(i) = g_2(i) * e^{j2\pi F(i-2\sigma-1)sin(theta)}$

3. The GEF arrays h_1 and h_2 are flipped by the flip() function of MATLAB and stored in the arrays $h1_{flipped}$ and $h2_{flipped}$

4. For every pixel with indices (i,k) in the processing region of the Gabor filter such that $i > 2\sigma$ and $i \leq M - 2\sigma$ and $j > 2\sigma$ and $j \leq N - 2\sigma$

a) A flag count is set to $2\sigma + 1$.

b) For each value of k, when l varies from $i - 2\sigma$ to $i + 2\sigma$,

i) Compute $i_1(x, y)$ as

$i_1(i, k) = i_1(i, k) + im(l, k) * h1_{flipped}(\text{count} - 2\sigma)$

ii) The count flag is incremented by 1.

c) The pixels that are not processed by this loop are set to 0 as $i_1(x, y)$ was initialized to all zeros.

5. For every pixel with indices (i,k) in the processing region of the Gabor filter such that $i > 2\sigma$ and $i \leq M - 2\sigma$ and $j > 2\sigma$ and $j \leq N - 2\sigma$

a) A flag count is set to $2\sigma + 1$.

b) For each value of i, when l1 varies from $k - 2\sigma$ to $k + 2\sigma$,

i) Compute $i_2(x, y)$ as

$i_2(i, k) = i_2(i, k) + i_1(i, l1) * h2_{flipped}(\text{count} - 2\sigma)$

ii) The count flag is incremented by 1.

c) The pixels that are not processed by this loop are set to 0 as $i_2(x, y)$ was initialized to all zeros.

6. The absolute value of $i_2(x, y)$, i.e. $|i_2(x, y)|$ is calculated by the abs() function of MATLAB and is stored in the variable M_gaussian

7. M_gaussian is scaled to M_gaussian_scaled as

$\text{M\_gaussian\_scaled} = \frac{\text{M\_gaussian} - \text{minimum(M\_gaussian)}}{\text{maximum(M\_gaussian)} - \text{minimum(M\_gaussian)}}$

where minimum(M_gaussian) is the minimum value in the matrix M_gaussian which is 0 in this case due to non-processed pixels being considered as zero. maximum(M_gaussian) is the maximum value in the matrix M_gaussian.

The following functions are called:

gaussian.m (This function is a manual implementation of the Gaussian function)

The following functions are calling:

Demo_Q1.m,Demo_Q2.m and Demo_Q3.m

The gaussian function has been executed by the MATLAB code gaussian.m. The pseudocode of the Gaussian function is as follows:

Objective: To perform the Gaussian function with input x or y and the standard deviation σ.

Input Parameters:
x or y: the input argument to the Gaussian function. If x is passed to the function y is 0 and vice-versa.
sigma: standard deviation of the Gaussian function for Gabor filter

Returned Result:
val - Value computed after applying the Gaussian function

Processing Flow:

1. Compute val as follows:
   $val = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big\{ -\frac{1}{2}\Big(\frac{x^2+y^2}{\sigma^2}\Big)\Big\}$

No MATLAB functions are called:

The following functions are calling:

GEF_3.m: This MATLAB code implements the Gabor filter and the smoothing filter.

2.2 Smoothing Filter for Texture Segmentation

The smoothing filter is applied for post-processing the filtered image m(x,y) from Gabor filter to enable easier segmentation of the image in order to clearly identify the textures. The input to the smoothing filter is the image m(x,y) and by using the Gaussian function g(x,y), the output image $m^{'}(x,y)$ is obtained. The Mathematical representation for the smoothing filter is as follows:

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

Now, $g^{'}(x,y)$ can be represented as $g(x).g(y)$, as defined in equation 3. g(x) and g(y) have been represented

in equations 6 and 7 respectively. Due to this representation, the 2-D convolution operation has been executed by first applying the 1-D convolution for each y in m(x,y) with g_1 gaussian function in terms of x to generate an intermediate $m_1(x, y)$. Again a 1-D convolution is applied for each x in $m_1(x, y)$ with g_2 gaussian function in terms of y to generate the final smoothed image $M\_gaussian\_smooth(x, y)$

The smoothing filter has been implemented in the MATLAB code GEF_3.m after the Gabor filter.

Theoretically, the pseudocode of the smoothing filter is as follows:

Objective: To perform on image M_gaussian_tmp(x,y) with Gaussian function g(x,y).

Input Parameters:
M_gaussian_tmp: Filtered image after Gabor filter and scaling the image.
sigma: standard deviation of the Gaussian function for Gabor filter

Returned Result:
M_gaussian_smooth - Resulting image after applying the smoothing filter.

Processing Flow:

1. Set two arrays full of zeros with the same dimension (M,N) as the input image to denote $m_1(x, y)$ and $M\_gaussian\_smooth(x, y)$.
2. For each iteration i in the window of $4\sigma + 1$
   a) Compute the Gaussian function g_1 where x is $i - 2\sigma - 1$
   b) Compute the Gaussian function g_2 where y is $i - 2\sigma - 1$
3. The GEF arrays g_1 and g_2 are flipped by the flip() function of MATLAB and stored in the arrays $g1_{flipped}$ and $g2_{flipped}$
4. For every pixel with indices (i,k) in the processing region of the Gabor filter such that $i > 2\sigma$ and $i \leq M - 2\sigma$ and $j > 2\sigma$ and $j \leq N - 2\sigma$
   a) A flag count is set to $2\sigma + 1$.
   b) For each value of k, when l varies from $i - 2\sigma$ to $i + 2\sigma$,
      i) Compute $m_1(x, y)$ as
      $m_1(i, k) = m_1(i, k) + M\_gaussian(l, k) * g1_{flipped}(\text{count} - 2\sigma)$
      ii) The count flag is incremented by 1.

c) The pixels that are not processed by this loop are set to 0 as $m_1(x, y)$ was initialized to all zeros.

5. For every pixel with indices (i,k) in the processing region of the Gabor filter such that $i > 2\sigma$ and $i \leq M - 2\sigma$ and $j > 2\sigma$ and $j \leq N - 2\sigma$

a) A flag count is set to $2\sigma + 1$.

b) For each value of i, when l1 varies from $k - 2\sigma$ to $k + 2\sigma$,

i) Compute $M\_gaussian\_smooth(x, y)$ as

$M\_gaussian\_smooth = M\_gaussian\_smooth(i, k) + m_1(i, l1) * g2_{flipped}(\text{count} - 2\sigma)$

ii) The count flag is incremented by 1.

c) The pixels that are not processed by this loop are set to 0 as $M\_gaussian\_smooth(x, y)$ was initialized to all zeros.

The following functions are called:

gaussian.m (This function is a manual implementation of the Gaussian function)

The following functions are calling:

Demo_Q1.m,Demo_Q2.m and Demo_Q3.m

2.3 Code organization

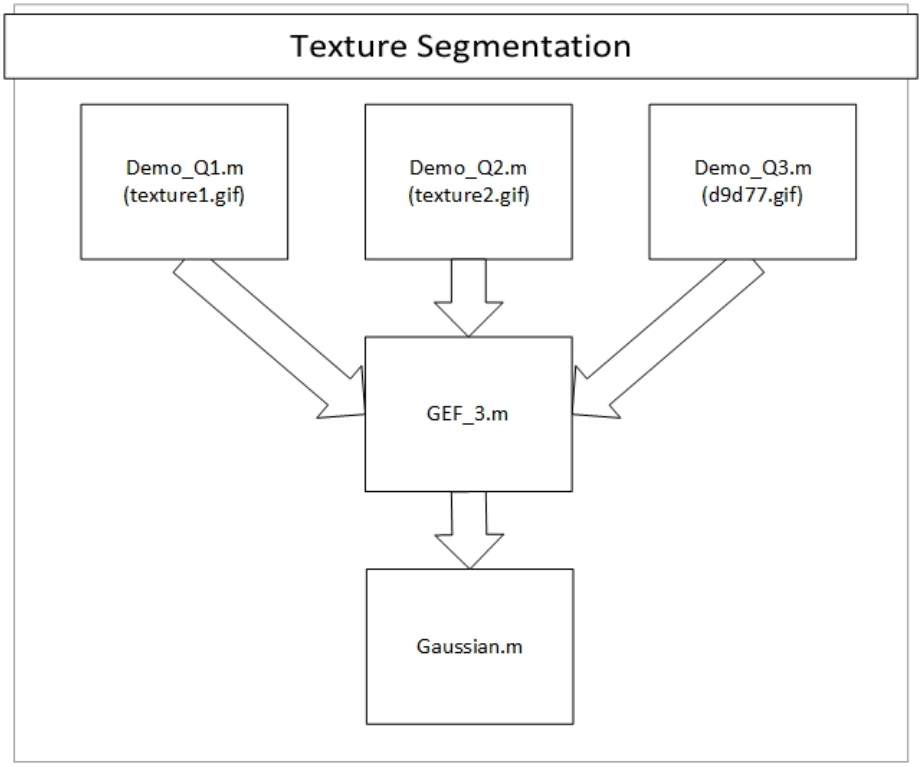

Figure 2: Overall Program Flow

The results for the images texture_1.gif and texture_2.gif have been implemented by the MATLAB codes Demo_Q1.m and Demo_Q2.m respectively while the d9d77.gif has been implemented by Demo_Q3.m. The images obtained after applying the Gabor filter have been thresholded and super-imposed on the original image to demonstrate the different textures in the segmented regions. A readme file README.md has been submitted which explains how to run the codes and obtain the results.

3 Results

3.1 Results for Image: "texture1.gif"

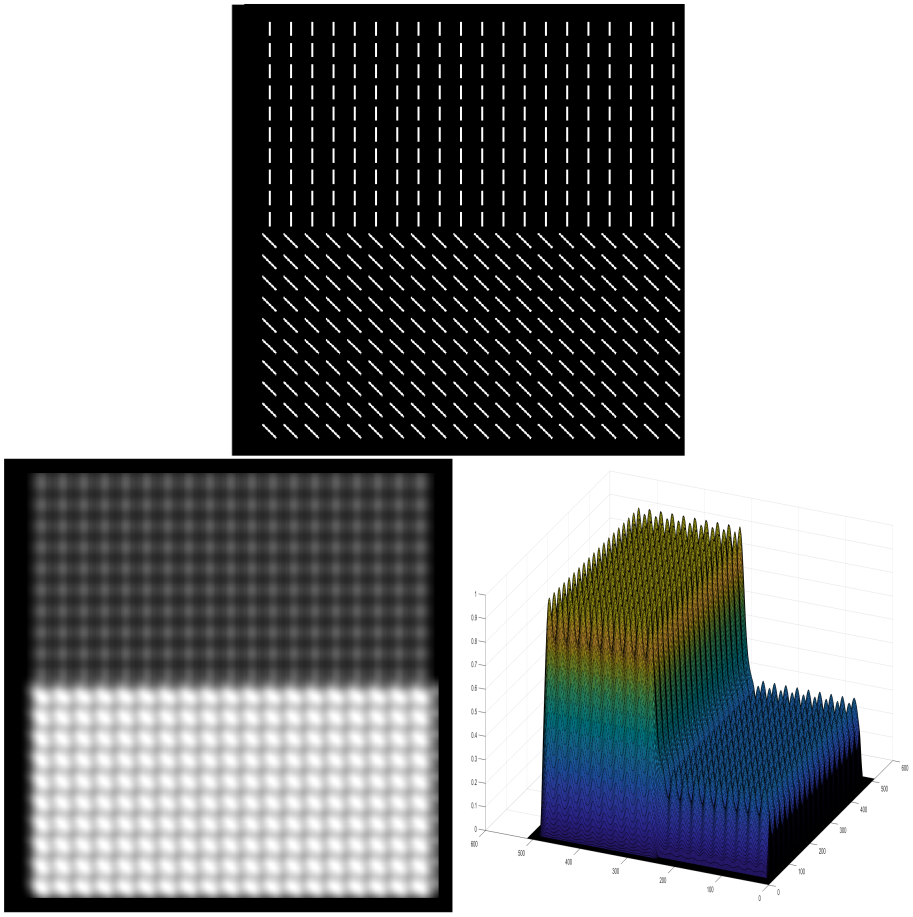

Figure 3: (a) Original Image $I(x, y)$ (b) Filtered Image using Gabor Filter $m(x, y)$ (c) Surface plot of Gabor Filtered Image

Figure 3 (a) shows the Original Image $I(x, y)$. On observing, we can see that the image has two distinct line types which are the different textures. Figure 3 (b) is the output of the Gabor Texture segmentation filter on the image. **The filter successfully segments the two re-**

gions by indicating them with different grayscale intensities. The top texture region has an average lower grayscale intensity than the bottom texture region. The parameters used for this image are as follows:

- $\theta = 135^{\circ}$
- $\sigma = 8$
- $F = 0.059$ cycles/pixel
- $\sigma_{smooth} = 24$

Figure 3 (c) shows the surface plot with x and y axis as the pixel coordinates and z axis is the grayscale intensity values given by the filtered image $m(x, y)$.

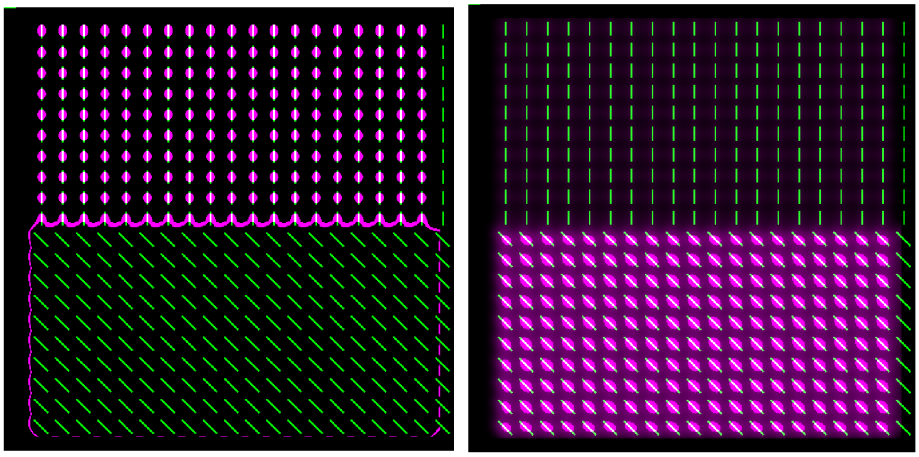

Figure 4: (a) Thresholded $m(x, y)$ overlaid on original image $I(x, y)$ (Top texture) (b) Thresholded $m(x, y)$ overlaid on original image $I(x, y)$ (Bottom texture)

Figure 4 shows the thresholded $m(x, y)$ overlaid on the original image. Figure 4 (a) shows the result for $threshold = (0.3,\ 0.4)$. We can observe from figure 4 (a) that the thresholded region represents the top texture.

But, there is a thin boundary extending below the actual texture region because of some outlier pixels. Figure 4 (b) shows the result for $threshold = [0.9,\ 1]$. We can observe from figure 4 (b) that the thresholded region represents the bottom texture. **This threshold range accurately represents the bottom texture region.**

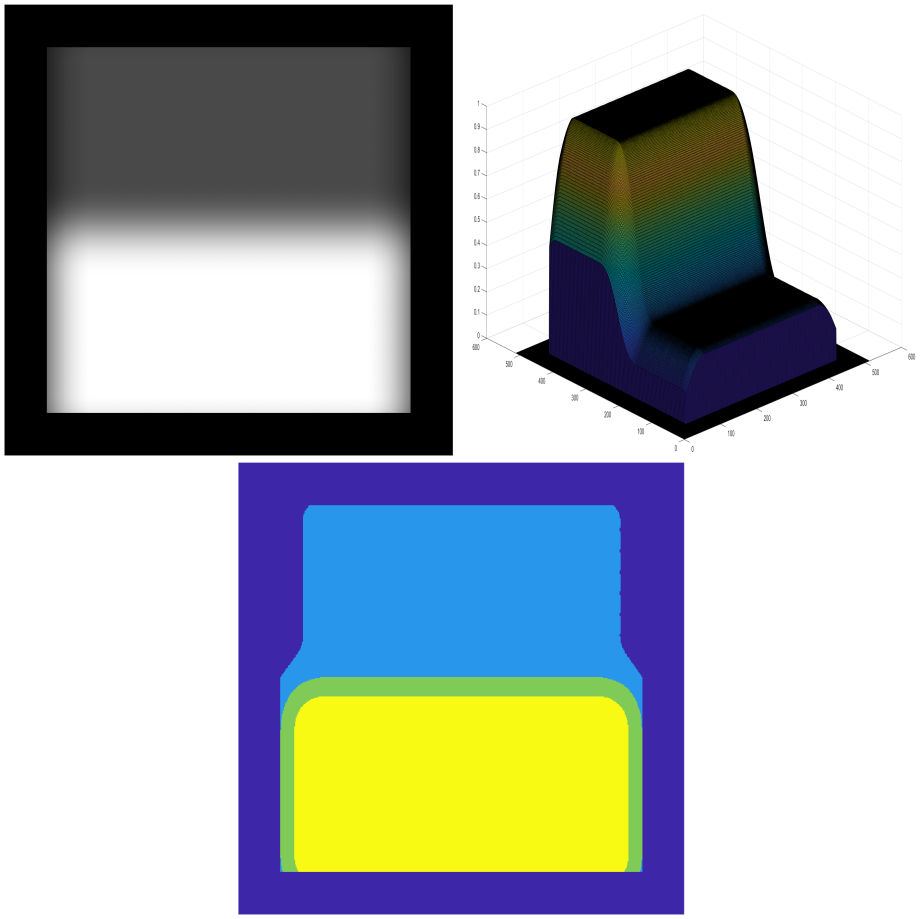

Figure 5: (a) Smoothed image using Smoothing Filter $m'(x, y)$ (b) Surface plot of Smoothed Image (c) Smoothed texture regions indicated by colormap

Figure 5 (a) shows the smoothed image $m'(x, y)$. The

smoothing operation is performed on the Gabor filter output $m(x, y)$ (fig 3b). Figure 5 (a) is an example of a successful smoothing operation. **After smoothing, the regions appear to be more distinct than fig 3b**. The surface plot in figure 5 (b) shows the surface plot with x and y axis as the pixel coordinates and z axis is the grayscale intensity values given by the smoothed image $m'(x, y)$. We can see that the surface plot is much smoother than one in figure 3 (c). Figure 5 (c) is the smoothed region represented as a colormap. The dark blue corresponds to the black background, the yellow corresponds to the bottom texture region and the light blue corresponds to the top texture region. The green color appears because of some stray pixels at the boundary of the bottom texture region. Also, the light blue region looks reduced because of some stray pixels at the boundary of the top texture region.

3.2 Results for Image: "texture2.gif"

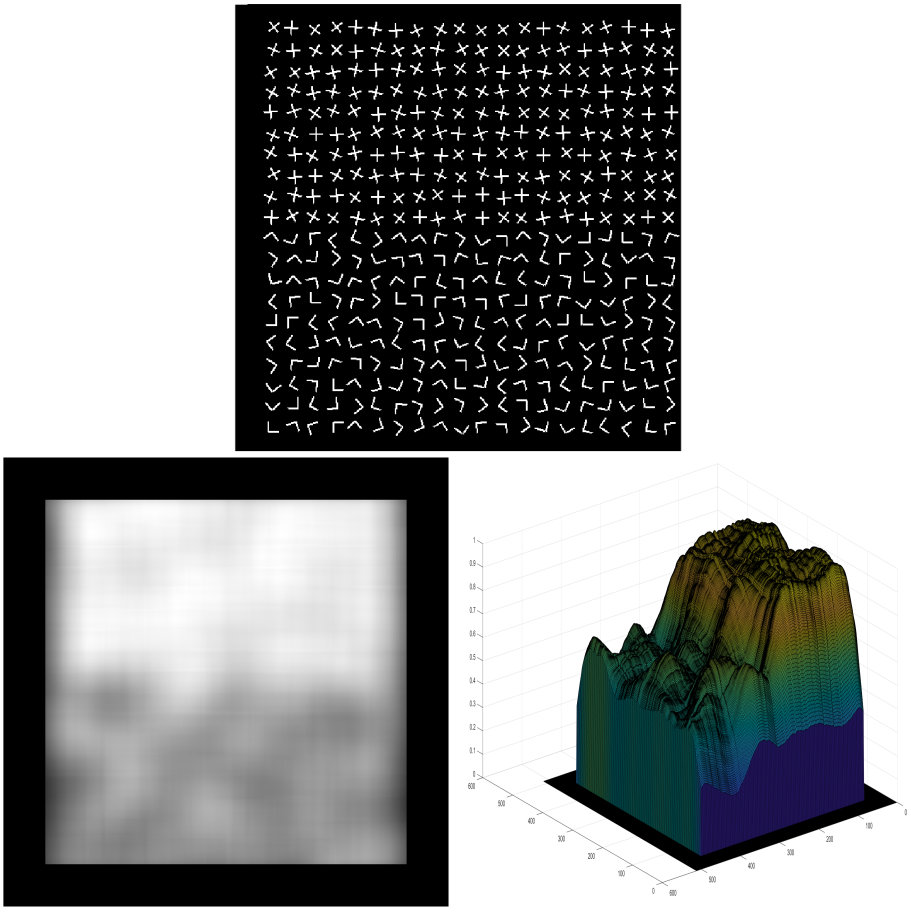

Figure 6: (a) Original Image $I(x, y)$ (b) Filtered Image using Gabor Filter $m(x, y)$ (c) Surface plot of Gabor Filtered Image

Figure 6 (a) shows the Original Image $I(x, y)$. On observing, we can see that the image has two distinct objects - '+' and 'L' with different orientations. Figure 6 (b) is the output of the Gabor Texture segmentation filter on the image. **The filter segments the two regions by indicating them with different grayscale intensities, but there are some bright patches within**

the dark region and some dark patches within the bright region. The top texture region has an average higher grayscale intensity than the bottom texture region. The parameters used for this image are as follows:

- $\theta = 0°$
- $\sigma = 24$
- $F = 0.042$ cycles/pixel
- $\sigma_{smooth} = 24$

Also, figure 6 (b) is much smoother than 3 (b) because we chose a higher sigma value here. Figure 6 (c) shows the surface plot with x and y axis as the pixel coordinates and z axis is the grayscale intensity values given by the filtered image $m(x, y)$.

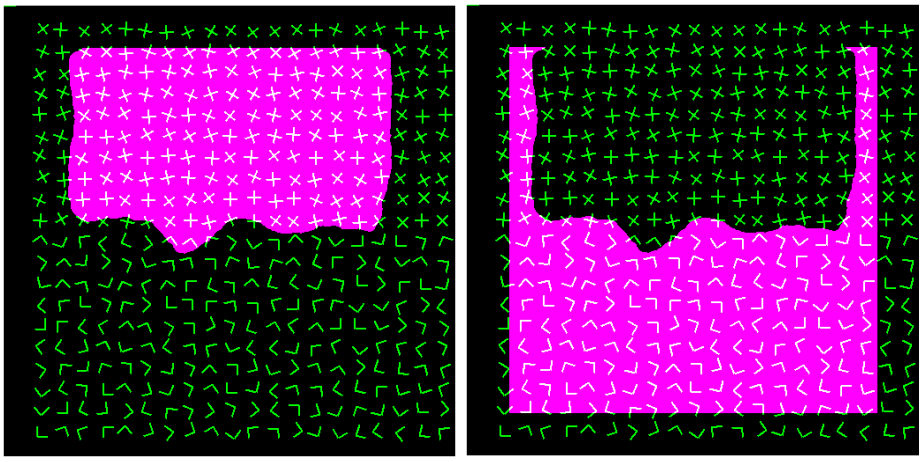

Figure 7: (a) Thresholded $m(x, y)$ overlaid on original image $I(x, y)$ (Top texture) (b) Thresholded $m(x, y)$ overlaid on original image $I(x, y)$ (Bottom texture)

Figure 7 shows the thresholded $m(x, y)$ overlaid on the original image. Figure 7 (a) shows the result for

threshold = [0.8, 1]. We can observe from figure 7 (a) that the thresholded region represents the top texture. This threshold range represents the top texture region. It appears that the bottom boundary of the top region includes some objects from the bottom texture region. **This may be because of the confusion caused by orientation changes. In some orientations, a '+' may be computed as being similar to an 'L' and vice versa**. Figure 7 (b) shows the result for *threshold* = (0, 0.8). We can observe from figure 7 (b) that the thresholded region represents the bottom texture. But, there is a thin boundary extending above the actual texture region because of some outlier pixels.

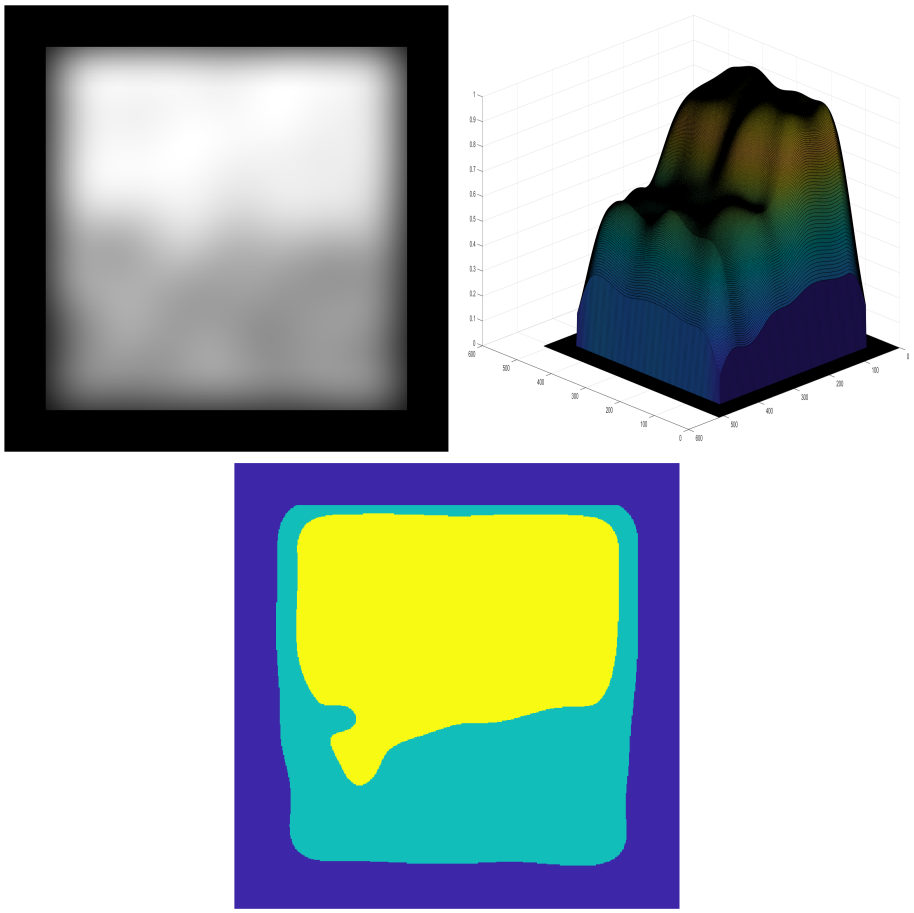

Figure 8: (a) Smoothed image using Smoothing Filter $m'(x, y)$ (b) Surface plot of Smoothed Image (c) Smoothed texture regions indicated by colormap

Figure 8 (a) shows the smoothed image $m'(x, y)$. The smoothing operation is performed on the Gabor filter output $m(x, y)$ (fig 6b). Figure 8 (a) is an example of a successful smoothing operation. After smoothing, the regions appear to be more distinct and than fig 6b. **It appears that the dark patches within the bright region and vice versa have been reduced by the smoothing operation**. The surface plot in figure 8 (b) shows the surface plot with x and y axis as the pixel

coordinates and z axis is the grayscale intensity values given by the smoothed image $m'(x, y)$. We can see that the surface plot is much smoother than one in figure 6 (c). Figure 8 (c) is the smoothed region represented as a colormap. The dark blue corresponds to the black background, the yellow corresponds to the top texture region and the light blue corresponds to the bottom texture region. The light blue region looks appears to extend to the top due to the presence of some outlier pixels at the boundary.

3.3 Results for Image: "d9d77.gif"

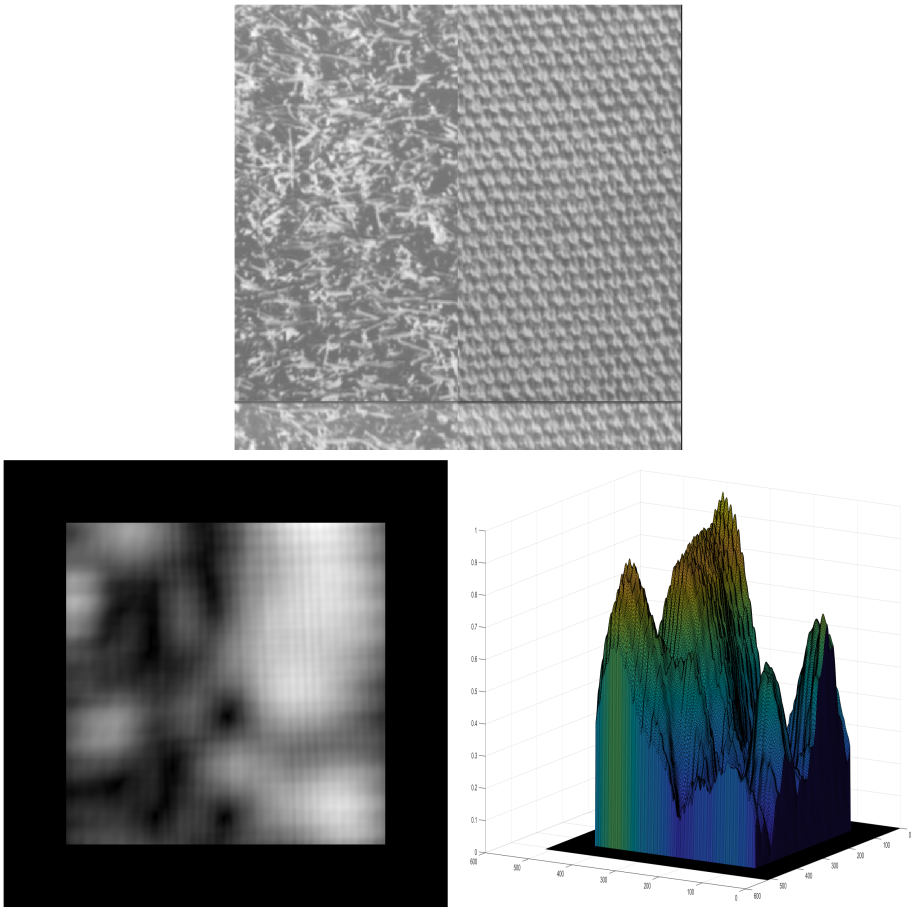

Figure 9: (a) Original Image $I(x, y)$ (b) Filtered Image using Gabor Filter $m(x, y)$ (c) Surface plot of Gabor Filtered Image

Figure 9 (a) shows the Original Image $I(x, y)$. On observing, we can see that the image has two distinct textures - lawn-grass (weakly ordered) and cotton-canvas (strongly ordered). Figure 9 (b) is the output of the Gabor Texture segmentation filter on the image. **The filter segments the two regions by indicating them with different grayscale intensities, but there are many**

bright patches within the dark region and some dark patches within the bright region. The occurrence of contrasting intensity patches are more than in 6(b). The left texture region has an average lower grayscale intensity than the right texture region. The parameters used for this image are as follows:

- $\theta = 60^{\circ}$
- $\sigma = 36$
- $F = 0.063$ cycles/pixel

Also, figure 9 (b) is much smoother than 3 (b) and 6 (b) because we chose a higher sigma value here. Figure 9 (c) shows the surface plot with x and y axis as the pixel coordinates and z axis is the grayscale intensity values given by the filtered image $m(x, y)$.

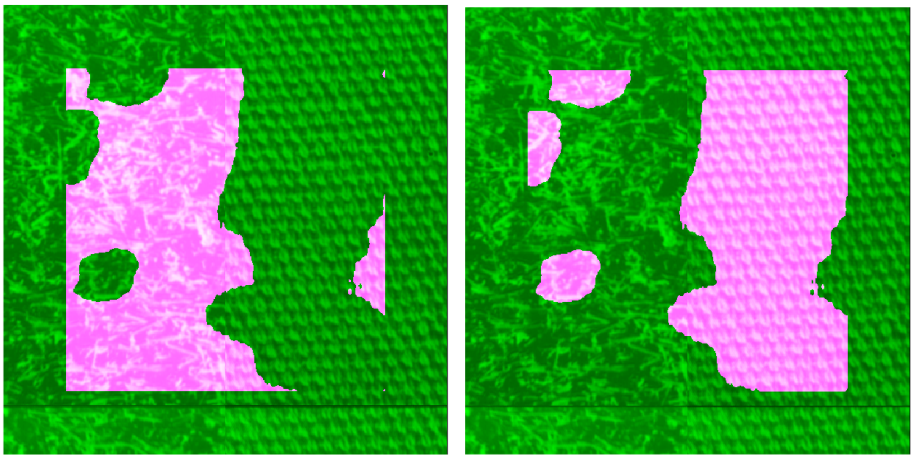

Figure 10: (a) Thresholded $m(x, y)$ overlaid on original image $I(x, y)$ (Left texture) (b) Thresholded $m(x, y)$ overlaid on original image $I(x, y)$ (Right texture)

Figure 10 shows the thresholded $m(x, y)$ overlaid on the original image. Figure 10 (a) shows the result for

threshold = (0, 0.4]. We can observe from figure 10 (a) that the thresholded region represents the left texture. This threshold range represents the left texture region. Figure 10 (b) shows the result for *threshold* = (0.4, 1]. We can observe from figure 10 (b) that the thresholded region represents the right texture. **The results from thresholding shows that the number of bright patches within the darker region is more than the number of dark patches in the brighter region. The former may be due to some strongly ordered patches within the lawn-grass and the latter may be due to the influence of some boundary pixels.**

3.4 Coordinate Limits of the Processed Images

Not all pixels of the image are processed as the window of the GEF in Gabor filter is in the range of $[-2\sigma, 2\sigma]$ where σ is the standard deviation of the Gaussian function for the Gabor filter. So, for any 2-D image with dimensions, MxN, the portion of the i(x,y) that are actually segmented are the pixel values that satisfy the conditions $x > 2\sigma$ and $x \leq M - 2\sigma$ and $y > 2\sigma$ and $y \leq M - 2\sigma$. The same condition is applied for the smoothing filter on the filtered image after Gabor Filter, which is the input to the smoothing filter. The same condition is applied as the window is in the range of $[-2\sigma, 2\sigma]$ where σ is the standard deviation of the Gaussian function for the smoothing filter. The unprocessed pixels are treated as zero for the Gabor filter and the smoothing filter.

For texture1.gif image with size 512x512, the σ val-

ues are 8 for Gabor filter and 24 for smoothing filter. Therefore if the image is represented as i1(x,y), then the coordinate limits for the image that can be processed by the Gabor filter is $x > 16$ and $x \leq 496$ and $y > 16$ and $y \leq 496$. If the gabor filter generates the image m1(x,y), then the coordinate limits for the image that can be processed by the smoothing filter is $x > 48$ and $x \leq 464$ and $y > 48$ and $y \leq 464$.

For texture2.gif image with size 512x512, the σ values are 24 for Gabor filter and 24 for smoothing filter. Therefore if the image is represented as i2(x,y), then the coordinate limits for the image that can be processed by the Gabor filter is $x > 48$ and $x \leq 464$ and $y > 48$ and $y \leq 464$. If the gabor filter generates the image m2(x,y), then the coordinate limits for the image that can be processed by the smoothing filter is $x > 48$ and $x \leq 464$ and $y > 48$ and $y \leq 464$.

For d9d77.gif image with size 512x513, the σ values are 36 for Gabor filter and 24 for smoothing filter. Therefore if the image is represented as i3(x,y), then the coordinate limits for the image that can be processed by the Gabor filter is $x > 72$ and $x \leq 441$ and $y > 72$ and $y \leq 441$. If the gabor filter generates the image m3(x,y), then the coordinate limits for the image that can be processed by the smoothing filter is $x > 48$ and $x \leq 465$ and $y > 48$ and $y \leq 465$.

4 Conclusions

The Gabor filter is useful to distinguish the different textures in the texture_1.gif, texture_2.gif and d9d77.gif. However, the quality of segmentation after thresholding the filtered image improves on application of the smoothing filter which smooths the filtered output from Gabor filter. The 3D plots of the filtered image and the smoothed image have a set of peaks at two different heights denote the 2 different types of textures in each image.

The best quality of segmentation is observed for the texture_1.gif image, followed by the texture_2.gif image and then the d9d77.gif image. This is due to selection of a lower σ value for texture_1.gif image over the other images and also due to distribution of the pixels in each of the images. The uniformity of the texture objects is also a key factor in the segmentation. Highly varying orientations and disordered textures can influence the effectiveness of the segmentation process.