## CSE585/EE555:  Digital Image Processing II

## Computer Project # 4:

## Texture Segmentation

#### Rohan Shah, Thomas Jarvinen

#### Date: 4/6/2018

* + 1. **Objectives**

This project aimed to demonstrate simple segmentation of textures using Gabor filters in conjunction with Gaussian smoothing via 2-dimensional convolution.

* + 1. **Methods**

In this project, image segmentation was realized through 5 processes. Gabor filters and circular Gaussian filters were approximated by computing 2-dimensional masks covering 2 standard deviations of the pdf in x and y dimensions. Using these masks as an input, filtering was then realized via 2-dimensional discrete convolution. The output of these results was then normalized to vary from values 1 to 255 so they could be viewed in grayscale. Finally, thresholding was performed on the resulting images to find a final segmentation

*1. Circular symmetric Gaussian filter*

Circular symmetric Gaussian filtering was approximated using a real-valued 2-dimensional discrete mask of dimension 4σ +1 x 4σ +1. This approximation covers two standard deviations along all dimensions of the distribution, or about 95% of relevant values. To compute this mask, a simple nested for loop was used to compute all values of g(x,y) for integer values of x,y in the range -2 σ < x < 2 σ; -2 σ < y < 2 σ



*2. Gabor filter*

To compute the complex Gabor mask or GEF, the following function was implemented:



This implementation takes F, σ, and θ as inputs, and computes a complex-valued 2-D mask in the range -2 σ < x < 2 σ; -2 σ < y < 2 σ using nested for loops in the same manner as above.

*3. 2-dimensional convolution*

2-dimensional discrete convolution is the computation of the weighted sum of products between a mask and an image in a sliding window fashion. That is, given image I(x,y) and mask (kernel) k(x,y)

This is implemented as a 2-dimensional for loop, which calculates the above formation at every possible position of the mask in the image. Note that the above formula requires a local window of width 2n+1 to be defined, so values to close to the edge of the image are automatically zeroed out.

*4. Display the result*

The results of the Gabor filter need to be scaled to the range of 0 to 255 for visualization purposes. To do so, the real portion of the Gabor filter output is extracted. Then the minimum value in the range is subtracted from all values in the image. Then all values in the image are divided by (maxImageValue – minImageValue). This gives an image scaled to the range 0-255, which is then cast to the uint8 data type for viewing.

*5. Segmentation*

**How to run code**

* + 1. **Results and Discussion**
    2. **Conclusions**