Lecture 19

Feature-based Texture Classification

ECEN5283 Computer Vision

Dr. Guoliang Fan

Oklahoma State University

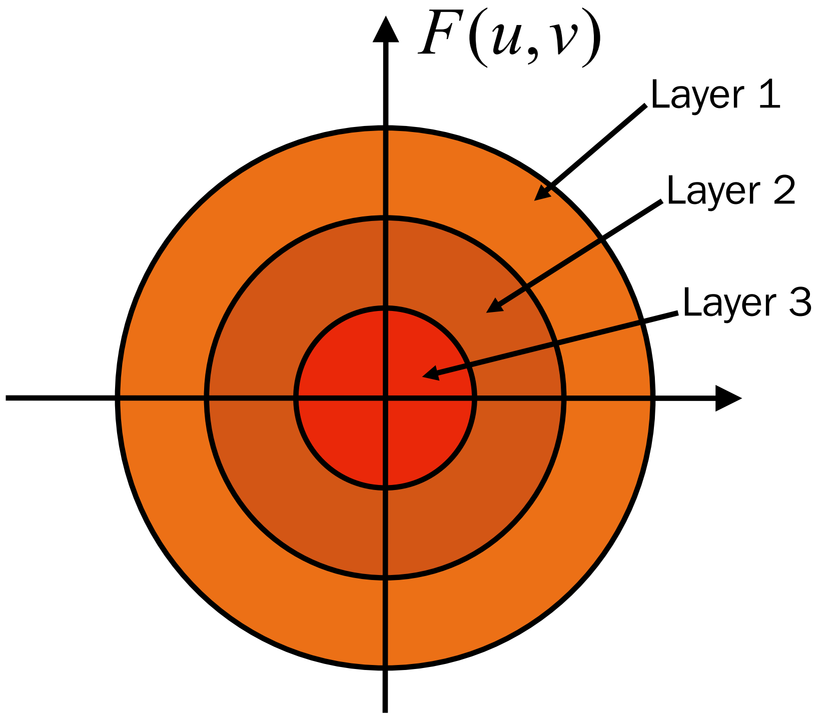
Goals

To review the Gaussian and Laplacian pyramids for multiscale image representation.

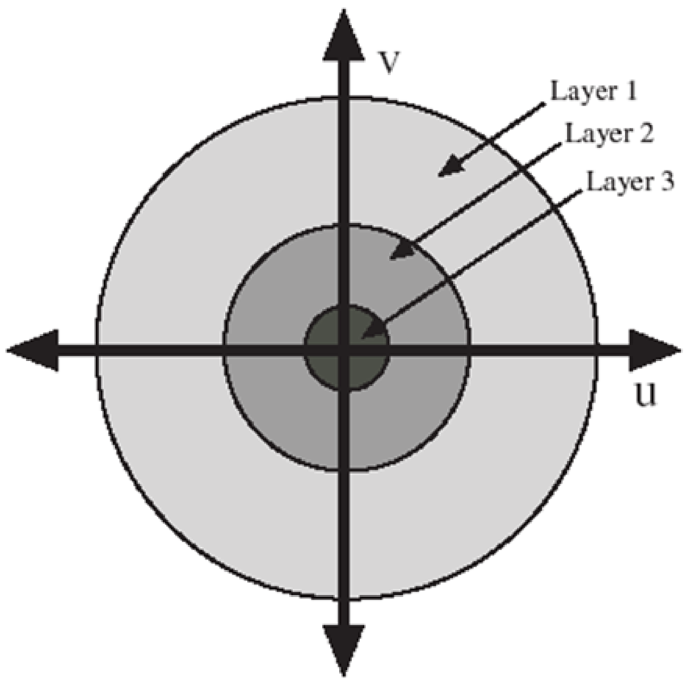
To develop a feature-based technique of texture classification

To introduce Project 3 on texture classification

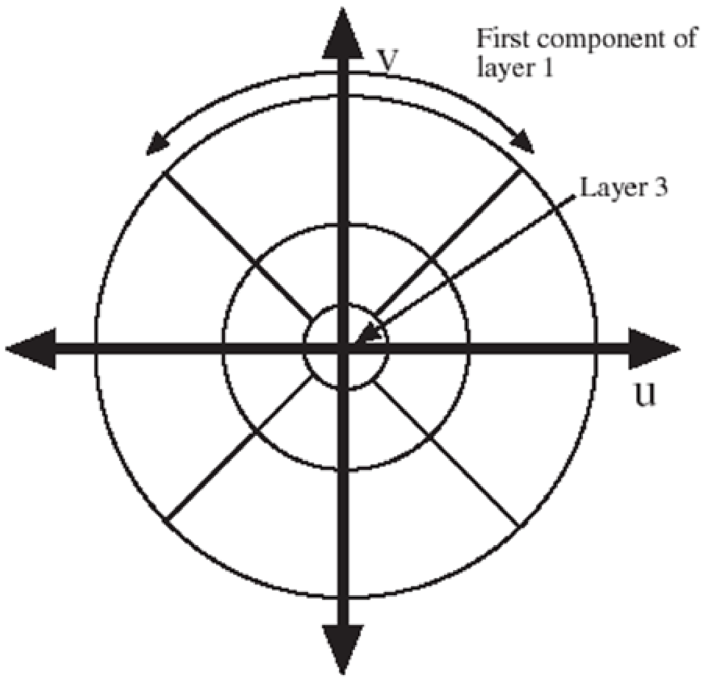
Oriented and Non-oriented Multiscale Image Representation in the Frequency Domain



Gaussian Pyramid

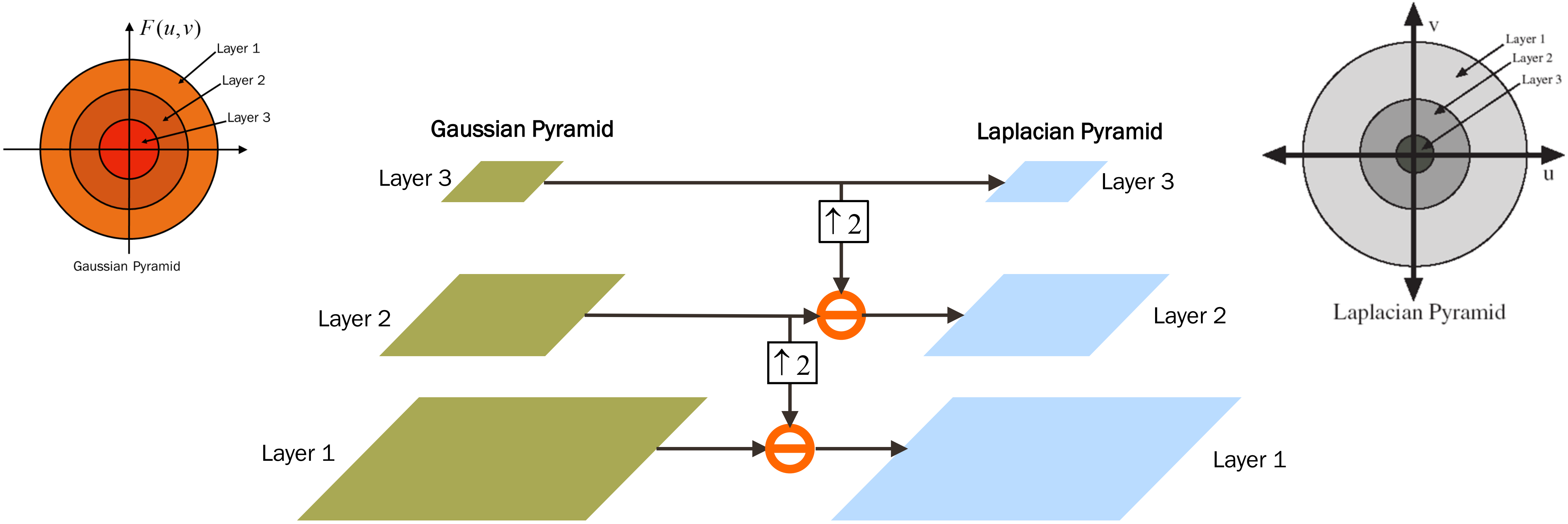


Laplacian Pyramid

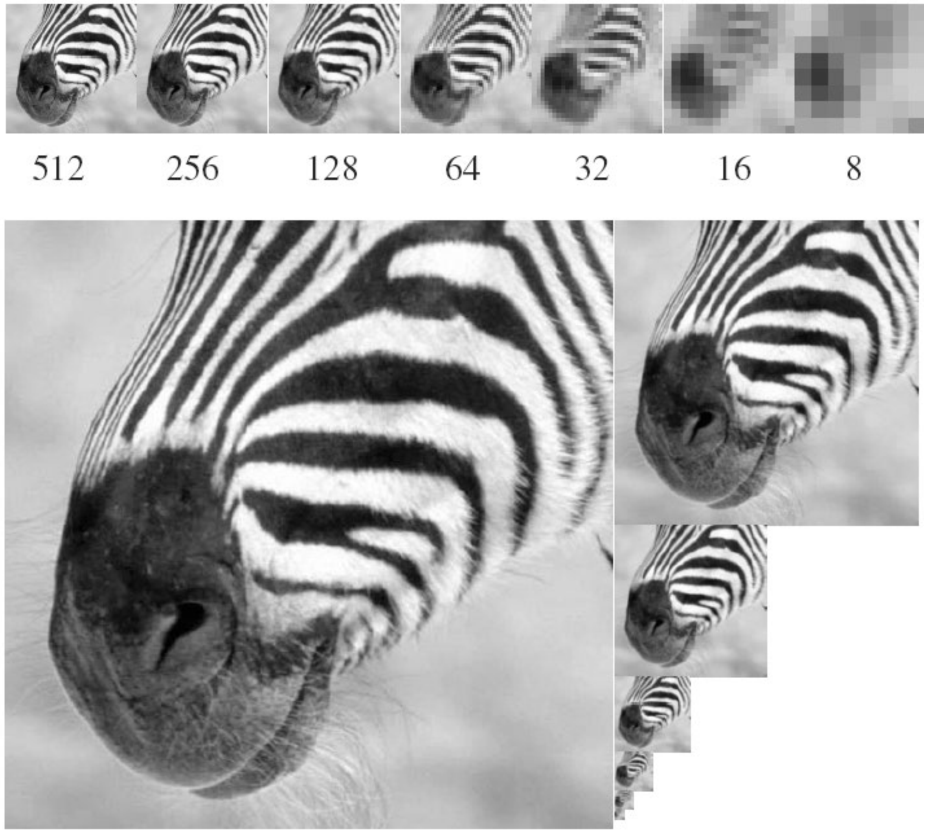


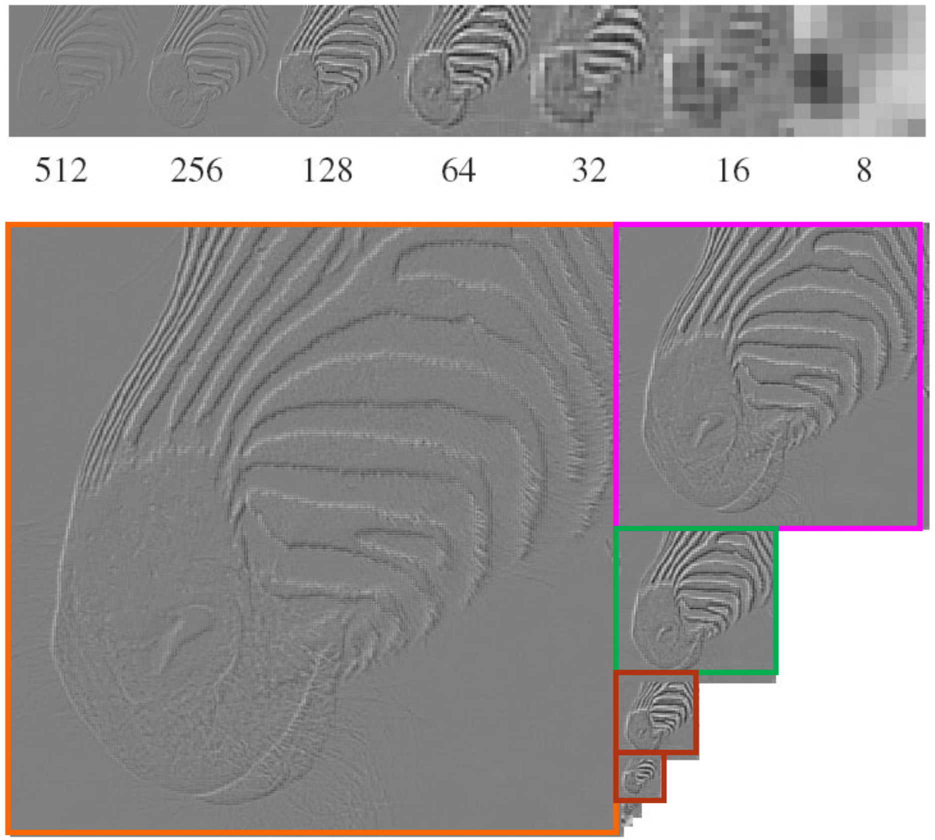
Oriented Pyramid

Gaussian and Laplacian Pyramids

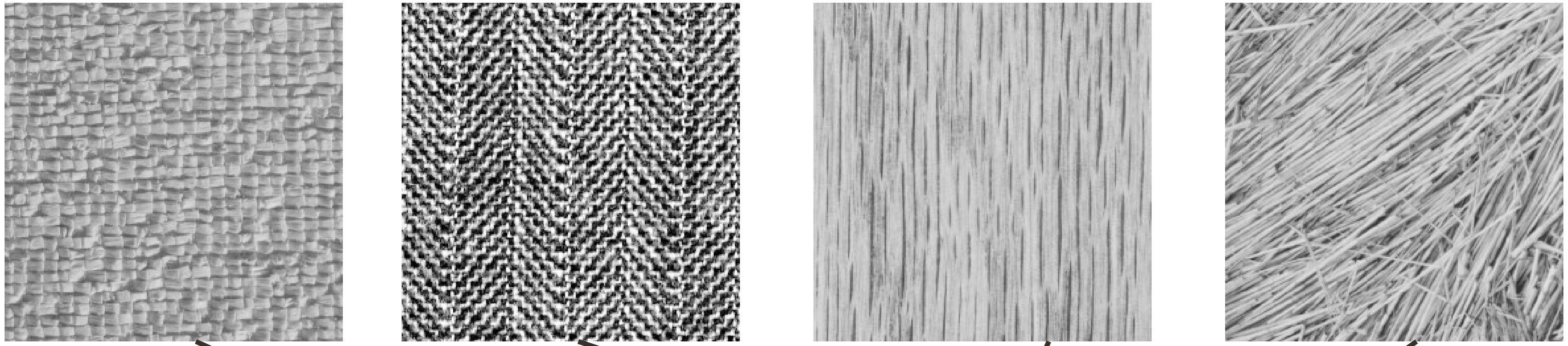


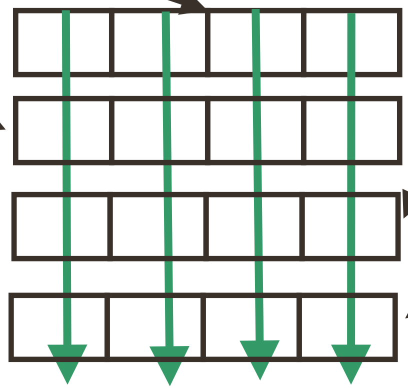
Laplacian-based Texture Analysis





Texture Classification: Training





$$\left(\begin{matrix} v_1^{\max} \\ v_1^{\min} \end{matrix}\right)\left(\begin{matrix} v_2^{\max} \\ v_2^{\min} \end{matrix}\right)\left(\begin{matrix} v_3^{\max} \\ v_3^{\min} \end{matrix}\right)\left(\begin{matrix} v_4^{\max} \\ v_4^{\min} \end{matrix}\right)$$

$$V^{(i)} = \left\{v_1^{(i)}, v_2^{(i)}, v_3^{(i)}, v_4^{(i)}\right\}$$

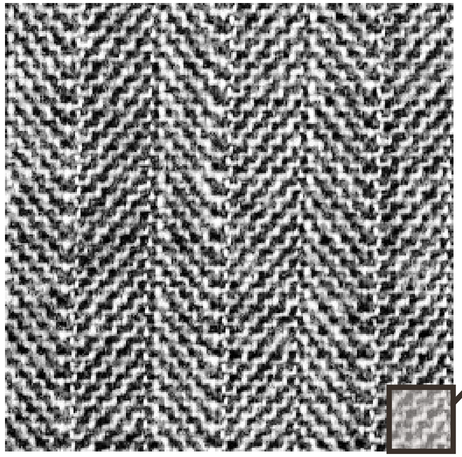
(original feature vectors for N textures, $i = 1, \ldots, N$)

All normalized feature vectors form a texture library

$$\bar{V}^{(i)} = \left(\frac{v_1^{(i)} - v_1^{\min}}{v_1^{\max} - v_1^{\min}}, \frac{v_2^{(i)} - v_2^{\min}}{v_2^{\max} - v_2^{\min}}, \ldots, \frac{v_4^{(i)} - v_4^{\min}}{v_4^{\max} - v_4^{\min}}\right)$$

Extreme value normalization

Texture Analysis: Classification



$$U = \{u_1, u_2, u_3, u_4\}$$

Feature normalization

$$\bar{U} = \left(\frac{u_1 - v_1^{\min}}{v_1^{\max} - v_1^{\min}}, \frac{u_2 - v_2^{\min}}{v_2^{\max} - v_2^{\min}}, \dots, \frac{u_4 - v_4^{\min}}{v_4^{\max} - v_4^{\min}}\right)$$

$$\text{Class}(\bar{U}) = \arg_{c\in\{1,2,..N\}} \min\left|\bar{V}^{(c)} - \bar{U}\right|$$

$$\bar{V}^{(i)} = \left\{\bar{v}_1^{(i)}, \bar{v}_2^{(i)}, \bar{v}_3^{(i)}, \bar{v}_4^{(i)}\right\} \ (i = 1, \dots, N)$$

Normalized feature vectors in the library

What is the major assumption for this classification scheme?

The assumption for this scheme is that the visual dissimilarity between two textures can be represented by the Euclidean distance of their respective feature vectors.

Project 3 Texture Classification

You are given 59 texture images for Project 3.

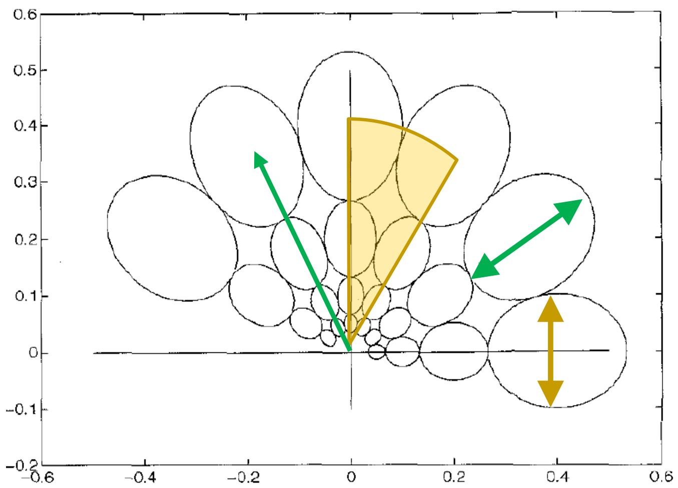
- First, obtain the feature vectors of different orders of statistics) for all textures (*make sure the matrices are converted to vectors first to call Matlab functions (mean, var, skewness, and kurtosis*).
- Normalize (using extremes) all feature vectors in each dimension across all textures, and save the normalized feature vectors in a texture library.
- For each texture image (640x640), divide it into 100 blocks of size 64x64, for which a normalized feature vector is computed.
- Classify all blocks by finding the closet feature vector in the texture library and compute the percentage of correct classification (PCC).

To test the Laplacian-based texture analysis method by using an open-source Matlab code of Laplacian pyramid and optimize the number of layers and the Gaussian filter.

To test Gabor filter-based texture analysis method by using the given Gabor filter Matlab function and optimize the numbers of layers and orientations.

Matlab Programming (1)
(Gabor filter Matlab function)

```
% GABORCONVOLVE - function for convolving image with log-Gabor filters
% EO = gaborconvolve(im,  nscale, norient, minWaveLength, mult, sigmaOnf, dThetaOnSigma)
%
%
%
%
%
%
%  Variable                Suggested   Description
%  name                    value
% ----------------------------------------------------------
%   im                                 Image to be convolved.
%   nscale                 = 4;        Number of wavelet scales.
%   norient                = 6;        Number of filter orientations.
%   minWaveLength          = 3;        Wavelength of smallest scale filter.
%   mult                   = 2;        Scaling factor between successive filters.
%   sigmaOnf               = 0.65;     Ratio of the standard deviation of the Gaussian describing the Gabor filter's
%                                      transfer function in the frequency domain to the filter center frequency.
%   dThetaOnSigma          = 1.5;      Ratio of angular interval between filter orientations and the standard
%                                      deviation of the angular Gaussian function used to construct filters in the
%                                      frequency plane.  A value of dThetaOnSigma = 1.5 results in approximately
%                                      the minimum overlap needed to get even spectral coverage.
```



http://www.csse.uwa.edu.au/~pk/research/matlabfns/

Matlab Programming (2)
(Multi-channel Gabor filtering output visualization)

```
clear all;
texture=imread('D7','bmp');
Ns=4;  No=6;

E0=gaborconvolve(texture,Ns,No,3,2,0.65,1.5);
% E0 is a cell that saves Ns X No Gabor filtering output images
% E0{i,j} is the output of the  ith scale and jth orientation.

for i=1:Ns
   for j=1:No
      ind=(i-1)*No+j;                 % Calculate the index of each sub-plot
      subplot(Ns,No,ind);             % Create a multi-figure plot
      imshow(abs(E0{i,j}),[]);        % Show the magnitude of each channel
   end
end
```

Matlab Programming (3)
(Read 59 Texture images)

```
Gau_Kernel=5;                          % the dimension of the smoothing kernel
Gau_Sigma=0.75;                        % the standard deviation of the smoothing kernel
Gau_Layer=4;                           % the layer number of the Laplacian pyramid
Texture_Num=59;                        % the total number of texture images
S=64;                                  % the block size for texture classification

Ti=cell(Texture_Num,1);                % to save all texture images
for i=1:Texture_Num
   N=num2str(i);                       % create the file name for each texture
   Ti{i}=imread(['D',N,'.bmp']);       % read a texture image into a cell
   Fi(i,:)=Laplacian_Pyramid(Ti{i},Gau_Layer,Gau_Sigma,Gau_Kernel);
                                       % Laplacian feature extraction (find an open-source code)
End

for i=1:Gau_Layer                      % normalize each dimension of all feature vectors
   Max_Var(i)=max(Fi(:,i));            % find the maximum value in each dimension
   Min_Var(i)=min(Fi(:,i));            % find the minimum value in each dimension
   Ni(:,i)=(Fi(:,i)-Min_Var(i))/(Max_Var(i)-Min_Var(i)); % create the normalized feature library for all training
images
end
```