TDI2131 Digital Image Processing (Week 6)
Tutorial 5

Note: Some images used in this tutorial belong to the Image Processing Toolbox. Other additional images are included with the tutorial.

1. Filtering in Frequency Domain (with Ideal Filter)

   (a) An ideal lowpass filter is given by the following equation,
   \[ H(u,v) = \begin{cases} 
   1 & \text{if } D(u,v) \leq D_0 \\
   0 & \text{if } D(u,v) > D_0 
   \end{cases} \]
   which can be perceived as a binary matrix of a white, centered circle of radius \( D_0 \), with a black background. We have created this image in the previous tutorial, and it can produce with these lines of code:
   ```
   >> [x,y]=meshgrid(-128:127,-128:127);
   >> z=sqrt(x.^2+y.^2);
   >> c=(z<15);
   >> img = imread('cameraman.tif');
   >> imgF = fftshift(fft2(img));
   >> imgFr = log(1+abs(imgF));
   >> figure, imshow(mat2gray(imgFr));
   ```
   Of course, if you have your function to show your Fourier spectrum conveniently, go ahead and use it. It will save you a lot of typing time. Next, perform lowpass filtering by multiplying the transformed image matrix by the “circle” matrix (recall that “dot asterisk” is the Matlab syntax for element-wise multiplication of two matrices):
   ```
   >> lpf = imgF.*c;
   >> lpf1 = log(1+abs(lpf));
   >> figure, imshow(mat2gray(lpf1));
   ```
   Finally, you need to take the inverse transform of the filtered image to display the actual result of the image.
   ```
   >> lpfimg = ifft2(ifftshift(lpf));
   >> figure, imshow(mat2gray(abs(lpfimg)));
   ```
   Describe what you observe from the results.

   (b) The cutoff frequency, \( D_0 \) used was 15. Find out what is the “optimum” cutoff frequency that allows sufficient smoothing with minimal artifact (ringing) effect.
(c) As for a high-pass ideal filter, it is just the complement of the low-pass ideal filter. The equation is now
\[
H(u, v) = \begin{cases} 
0 & \text{if } D(u, v) \leq D_0 \\
1 & \text{if } D(u, v) > D_0 
\end{cases}
\]
and we expect to get a black, centered circle with a white background.

Modify your previous lines of codes to create a high-pass ideal filter, and then process the image to see the result.

(d) If we want to process another image of a different size from cameraman.tif, we need to create a filter which is of same size as the image. For example, to process moon.tif (537x358), we need a filter of size 537x358 so that matrix multiplication works.

\[
\begin{align*}
&>> [x \ y] = \text{meshgrid}(-179:178,-268:268); \\
&>> z = \sqrt{x.^2+y.^2}; \\
&>> c = (z<15); \\
&>> \text{figure, imshow(c)};
\end{align*}
\]

Go ahead and complete the process.

2. Filtering with Butterworth Filters

For simplicity and easy comparisons later, let's go back to using cameraman.tif.

(a) A Butterworth lowpass filter is given by the following equation,
\[
H(u, v) = \frac{1}{1+(D(u, v)/D_0)^{2n}}
\]
which is easy to create using similar steps we used earlier. The only difference is in the function used.

\[
\begin{align*}
&>> [x, y] = \text{meshgrid}(-128:127,-128:127); \\
&>> \text{bl} = 1./(1+(\sqrt{x.^2+y.^2}/15).^(2*2)); \\
&>> \text{figure, imshow(bl)};
\end{align*}
\]

This creates a Butterworth lowpass filter of size 256x256 with $D_0=15$ and order $n=2$.

Here, take note that
- All multiplications and divisions are element-wise, using .* or ./
- As usual, $D_0$ which is 15, controls the radial size of the filter, however in actual calculation, the radius is double its normal size due to the exponent 2.

Now, we want to write some general Matlab functions that will help us generate Butterworth filters (lowpass & highpass) of any sizes.
New!: Inserting comments right after the function header will enable help contents to be displayed (when function help is used).

```matlab
function out=lbutte(im,d,n)
% LBUTTER(IM,D,N) creates a low-pass Butterworth filter
% of the same size as image IM, with cutoff D, and order N
% Use:
% x=imread('cameraman.tif');
% l=lbutte(x,25,2);
% height=size(im,1);
% width=size(im,2);
[achsen]=meshgrid(-floor(width/2):floor((width-1)/2),-floor(height/2):
floor((height-1)/2);
out = 1./(1+(sqrt(x.^2+y.^2)/d).^(2*n));
```

(b) A Butterworth highpass filter is easy to generate, now that we have its lowpass filter.

```matlab
function out=hbutte(im,d,n)
% HBUTTER(IM,D,N) creates a high-pass Butterworth filter
% of the same size as image IM, with cutoff D, and order N
% Use:
% x=imread('cameraman.tif');
% l=hbutte(x,25,2);
% out = 1 - lbutte(im,d,n);
```

Proceed to test out the filter.

3. Filtering with Gaussian Filters
(a) A Gaussian lowpass filter is given by the following equation,

\[ H(u,v) = e^{-D^2(u,v)^2/2\sigma^2} \]

However, Matlab is kind enough to provide the function fspecial (Remember?) which has can generate predefined Gaussian filters.

```matlab
>> gl = mat2gray(fspecial('gaussian',256,10));
>> figure, imshow(gl);
```

The first parameter of fspecial denotes the filter type, Gaussian, the second parameter is the size of filter, the third parameter is the standard deviation of the Gaussian distribution (\(\sigma\)). Use the help function if you need to figure out how to use fspecial.

Note the use of the mat2gray function here. The fspecial function on its own produces a lowpass Gaussian filter with a very small maximum (because it always adjusts its output to keep the sum of all coefficients 1). Therefore, the need to rescale its values so that the maximum value is 1.
(b) Similarly like other filters, the Gaussian highpass filter is just the complement of its lowpass filter. You just need to subtract its lowpass filter from 1.

4. Experiments

Using the following images,

- enamel.tif
- lena.jpg (included with tutorial)

write a script/function to filter these images with the following filters, using the same cutoff frequency ($D_0$) for all:

(i) Ideal lowpass
(ii) Butterworth lowpass
(iii) Gaussian lowpass (assume $\sigma = D_0$)
(iv) Ideal highpass
(v) Butterworth highpass
(vi) Gaussian highpass (assume $\sigma = D_0$)

Display all the results. Compare and analyse.

Homework

1. Homomorphic Filtering

If you have an image that suffers from variable illumination (dark in some sections, light in others), we may wish to enhance the contrast locally and, in particular, to enhance the darker regions. Here's an interesting image (weird_moon.jpg, included with tutorial) which suffers from this kind of problem – certain well-illuminated regions, and other regions in shadow that are dark. Well, obviously this is a very weird looking moon!

Write a program to perform homomorphic filtering on this image, using a slightly-modified form of Gaussian High Pass Filter (also found in Lecture 6, Slide 39 or textbook),

$$H(u, v) = (\gamma_H - \gamma_L)[1 - e^{-cD^2(u,v)/D_0^2}] + \gamma_L$$

where constant $c$ controls the steepness of slope, and $\gamma_L$ and $\gamma_H$ are the low gain and high gain values (preferably $\gamma_L < 1$ and $\gamma_H > 1$ for better results). $\sigma = D_0$ is assumed.