EECE-5626 (IP&PR) : Homework-3 Solutions

Table of Contents

Problem-1: Sampling in Two-Dimensions

A continuous-space sinusoidal signal $f_c(x,y) = 3\cos(2.4\pi x + 2.6\pi y)$ is sampled at (ξ_x, ξ_y) frequencies in sam/mt to obtain the image $f(m,n)$. An ideal reconstruction is used on $f(m,n)$ to obtain the analog signal $\hat{f}_c(x,y)$. Note that $\xi_x = 1/\Delta x$ and $\xi_y = 1/\Delta y$.

(a) Sampling Rate-1

If $\xi_x = 2$ sam/mt and $\xi_y = 3$ sam/mt, determine $f(m,n)$ and $\hat{f}_c(x,y)$.

Solution: The sampling intervals are $\Delta x = 0.5$ and $\Delta y = 1/3$. Hence

$$\begin{aligned} f(m,n) &= f_c(x = m\Delta x, y = n\Delta y) = 3\cos(1.2\pi m + 2.6\pi n/3) \\ &= 3[\cos(1.2\pi m)\cos(2.6\pi n/3) - \sin(1.2\pi m)\sin(2.6\pi n/3)] \\ &= 3\begin{bmatrix}\cos\{(2-0.8)\pi m\}\cos(2.6\pi n/3) \\ -\sin\{(2-0.8)\pi m\}\sin(2.6\pi n/3)\end{bmatrix} \\ &= 3[\cos(0.8\pi m)\cos(2.6\pi n/3) + \sin\{0.8\pi m\}\sin(2.6\pi n/3)] \\ &= 3\cos(0.8\pi m - 2.6\pi n/3) \end{aligned}$$

Since both digital radian frequencies are within the $(-\pi, \pi]$ range, the ideal reconstruction for sinusoidal signals implies that (m,n) can be replaced by $(x\xi_x, y\xi_y)$ to obtain the continuous signal. Therefore,

$$\widehat{f}_c(x,y) = 3\cos(0.8\pi 2x - 2.6\pi 3y/3) = 3\cos(1.6\pi x - 2.6\pi y).$$

Clearly, there is an aliasing in the x-direction.

(b) Sampling Rate-2

If $\xi_x = 3$ sam/mt and $\xi_y = 2$ sam/mt, determine $f(m,n)$ and $\widehat{f}_c(x,y)$.

Solution: Using the procedure similar to part (a) above, we obtain

$$\begin{aligned} f(m,n) &= f(x = m\Delta x, y = n\Delta y) = 3\cos(2.4\pi m/3 + 1.3\pi n) \\ &= 3\cos(2.4\pi m/3 - 0.7\pi n) \end{aligned}$$

Hence after ideal reconstruction

$$\widehat{f}_c(x,y) = 3\cos(2.4\pi 3x/3 - 0.7\pi 2y) = 3\cos(2.4\pi x - 1.4\pi y).$$

which indicates aliasing in the y-direction.

(c) Sampling Rate-3

If $\xi_x = 3$ sam/mt and $\xi_y = 3$ sam/mt, determine $f(m,n)$ and $\widehat{f}_c(x,y)$.

Solution: Again, using the procedure similar to part (a) above, we obtain

$$f(m,n) = f(x = m\Delta x, y = n\Delta y) = 3\cos(2.4\pi m/3 + 2.6\pi n/3).$$

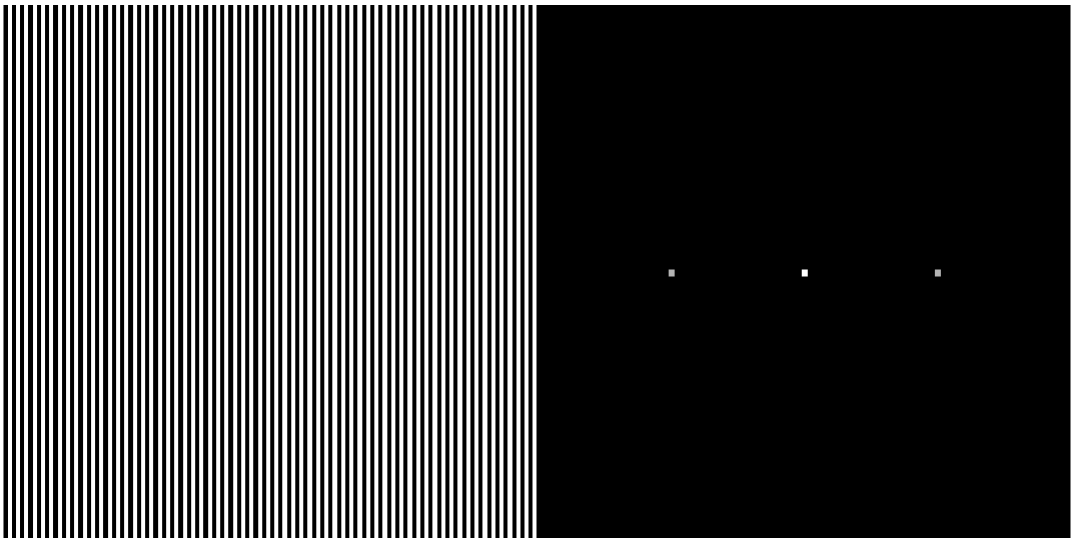
Hence after ideal reconstruction

$$\widehat{f}_c(x,y) = 3\cos(2.4\pi 3x/3 + 2.6\pi 3y/3) = 3\cos(2.4\pi x + 2.6\pi y).$$

Hence there is no aliasing.

Problem-2: DIP4E Problem 4.25, parts (b), (c), and (d) (Page 356)

The pair of images given in the book for this problem are difficult to see (especially, the one on the right). Use the following image pair to explain your answers.



```
% The following code generated the above two images
clc; clear; close all;
f = zeros(256); f(:,3:4:end) = 1; f(:,4:4:end) = 1;
F = fftshift(fft2(f)); Fm = abs(F);
h = fspecial('average',3); Fm = imfilter(Fm,h,'same');
imshowpair(f,Fm,'montage'); print -dpng ../artfiles/P4-28.png;
```

(b) Why Horizontal Spectrum?

Why are the components of the spectrum limited to the horizontal axis?

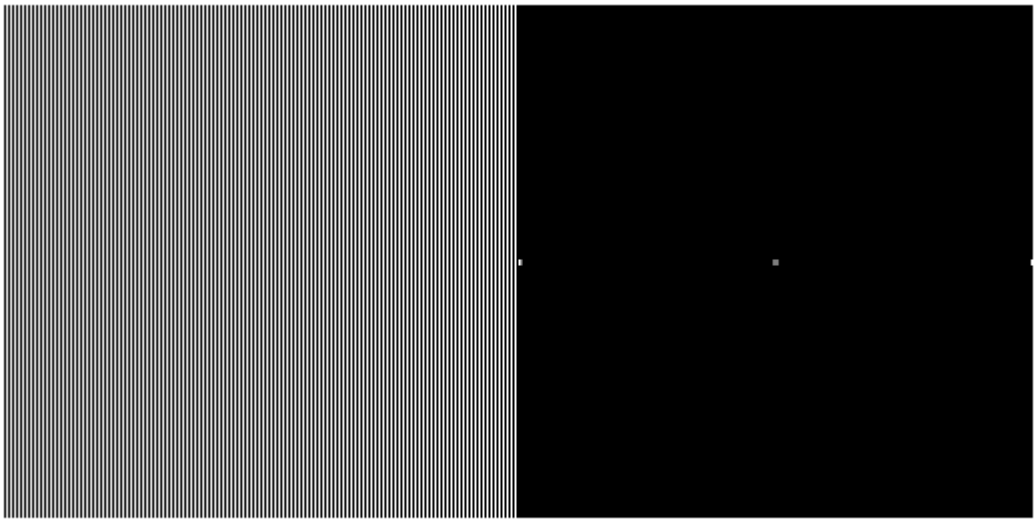
Solution: The image is constant along each column, which means that variations are limited only along each row, with each row being identical. Therefore, the DFT will have frequency components only along the horizotal axis. Along the vertical axis there is on a dc term, which is included as a part of the horizontal axis.

(c) New Spectrum

What would the spectrum look like for an image of the same size but having strips that are one pixel wide? Explain the reason for your answer.

Solution: We can arrive at the same solution in two different ways. The first is that one-pixel-wide stripes are the narrowest possible in a digital image. Therefore, the period will be the smallest possible period which implies the highest possible frequency. This means that the two spikes will appear at the furthest point on either side of the center. Another way to arrive at the same answer is to notice that, if the spikes of an image with two-pixel-wide stripes are midway between the center and ends of the horizontal axis of the spectrum, then the spikes of a signal with double the frequency must appear at twice the original distance. That is, the spikes of the signal with one-pixel-wide stripes must appear at the ends of the horizontal axis of the spectrum. The first answer is preferable because it shows a deeper understanding of the problem, but the second answer is acceptable in the context of how the problem statement was worded. The following MATLAB simulation demonstrates this.

```
f = zeros(256); f(:,2:2:end) = 1; %f(:,4:4:end) = 1;
F = fftshift(fft2(f)); Fm = mat2gray(abs(F));
h = fspecial('average',3); Fm = imfilter(Fm,h,'replicate');
Fm = [Fm,Fm(:,1)]; Fm = Fm/max(Fm(:));
imshowpair(f,Fm,'montage'); axis tight;
```



(d) DC Terms?

Are the dc terms in (a) and (c) the same, or are they different? Explain.

Solution: The dc term is equal to the average value of the signal. Because the images are of the same size and also contain an integer number of complete periods, the total areas occupied by black and white stripes are the same in both images. Therefore, their dc terms are the same.

Problem-3: DIP4E Problem 4.47 (Page 359)

The image on the right was obtained by: (a) multiplying the image on the left by $(-1)^{x+y}$; (b) computing the DFT; (c) taking the complex conjugate of the transform; (d) computing the inverse DFT; and (e) multiplying the real part of the result by $(-1)^{x+y}$. Explain mathematically why the image on the right appears as it does.

Solution: Note that according to the DIP4E book notation in Chapter-4, the variables, x, y, u, and v are discrete (integer) variables, with x and u in the range $[0, M-1]$ and y and v in the range $[0, N-1]$. See footnote on page-292.

The explanation is based on the following two properties of the DFT:

- $\mathcal{F}[f(x,y)(-1)^{x+y}] = F(u-M/2, v-N/2)$: See table 4.4 Property 4. This operation is nothing but the **`fftshift`** operation in MATLAB.
- $\mathcal{F}^{-1}[F^*(u,v)] = f(-x,-y)$: See Table 4.1 on page-292.

Using these two properties, we can now analyze the following sequence of operations:

1. $g(x,y) = f(x,y)(-1)^{x+y}$.
2. $G(u,v) = \mathcal{F}[f(x,y)(-1)^{x+y}] = F(u-M/2, v-N/2)$.
3. $W(u,v) = F^*(u-M/2, v-N/2)$.
4. $w(x,y) = \mathcal{F}^{-1}[F^*(u-M/2, v-M/2)] = (-1)^{-x-y} f(-x,-y)$, from 2.
5. $s(x,y) = (-1)^{-x-y} f(-x,-y)(-1)^{x+y} = f(-x,-y)$ which simply rotates $f(x,y)$ by $180°$, thus producing the image $s(x,y)$ on the right above.

Problem-4: DIP4E Project 4.6 (Page 363)

Lowpass filtering in the frequency domain

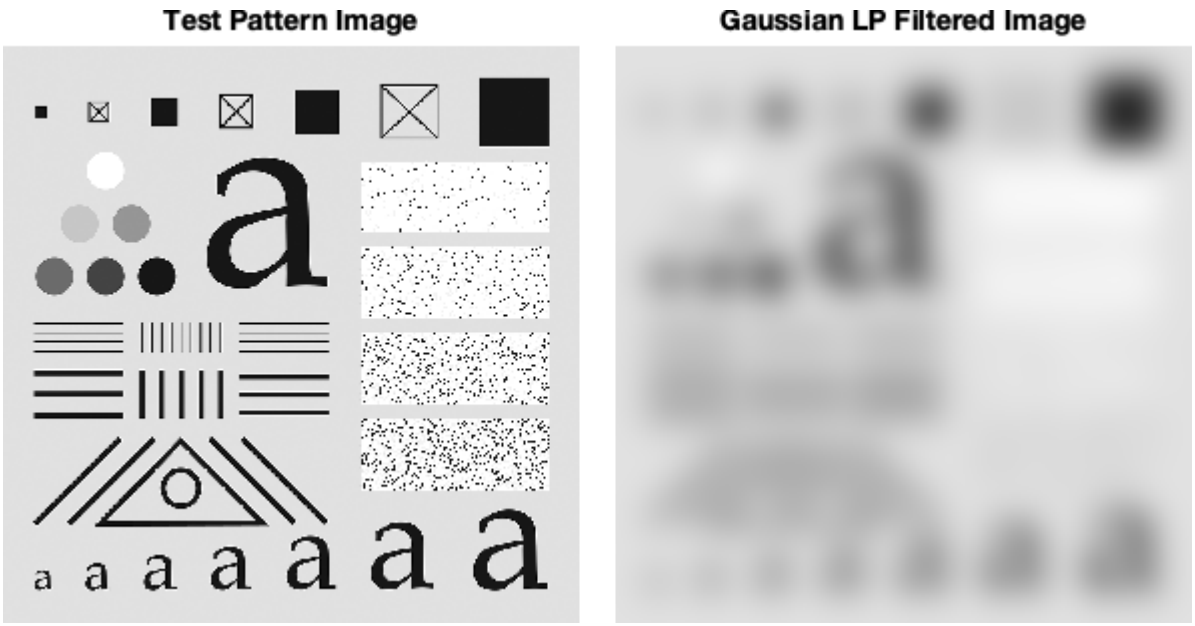
This problem uses the **`lpFilterTF4e`** function to generate a $P \times Q$ lowpass filter transfer function, which is described in DIP4E Project 4.3 (a) and the **`dftFiltering4e`** function from DIP4E Project 4.5 (a) to filter an image in the frequency domain.

(a) Gaussian Lowpass Filtering

Read the image **`testpattern1024.tif`** and lowpass filter it using a Gaussian filter so that the parge letter "a" is barely readable, and the other letters are not.

Solution: MATLAB script

```
clc; close all; clear;
f = imread('../artfiles/testpattern1024.tif');
[M,N] = size(f);
P = 2*M; Q = 2*N; % padding parameters;
% Gaussian filter
H = lpFilterTF4e('gaussian',P,Q,10); % The value of D0 = 10 determined experimentally
g = dftFiltering4e(f,H); % filtering in the frequency domain
figure('units','inches','Position',[0,0,8.25,4.25]);
subplot('position',[0,0,4/8.25,4/4.25]); imshow(f);
title('Test Pattern Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,4/4.25]); imshow(g);
title('Gaussian LP Filtered Image','FontSize',14);
```

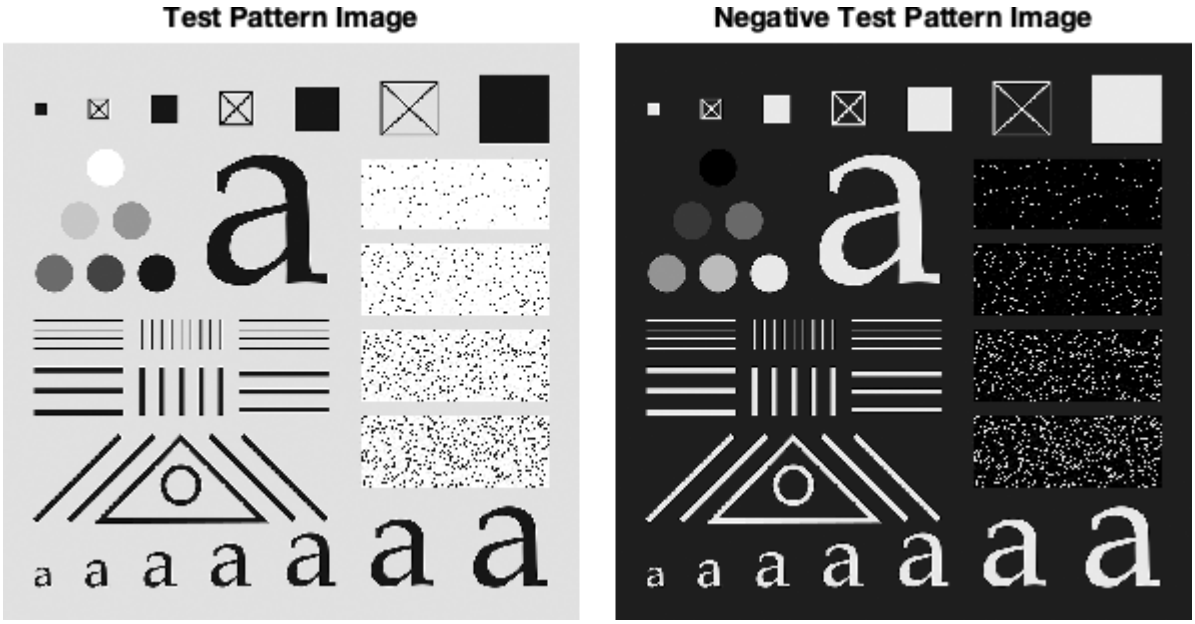


(b) Butterworth Lowpass Filtering and Thresholding

Read the image **`testpattern1024.tif`**. Lowpass filter it using a Butterworth filter of your specifications so that, when thresholded, the filtered image contains only part of the large square on the top, right.

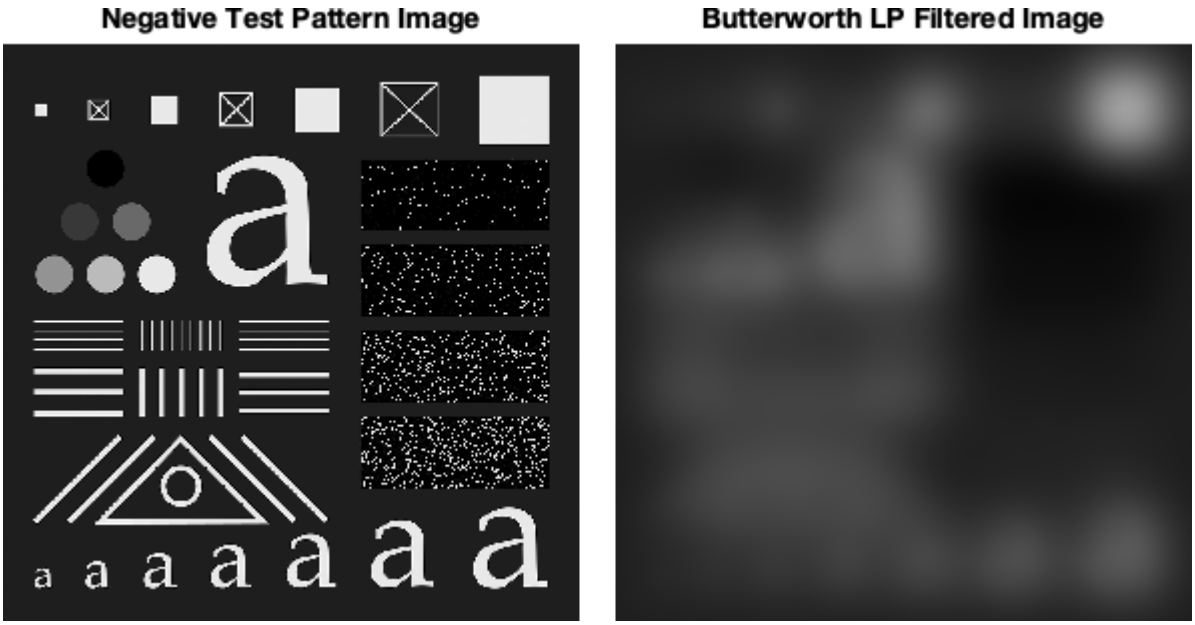
Solution: MATLAB script

```
% Negative of the function using DIP4E Project 3.2 (a) function
fn = intXform4e(f,'negative');
figure('units','inches','Position',[0,0,8.25,4.25]);
subplot('position',[0,0,4/8.25,4/4.25]); imshow(f);
title('Test Pattern Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,4/4.25]); imshow(fn);
title('Negative Test Pattern Image','FontSize',14);
```

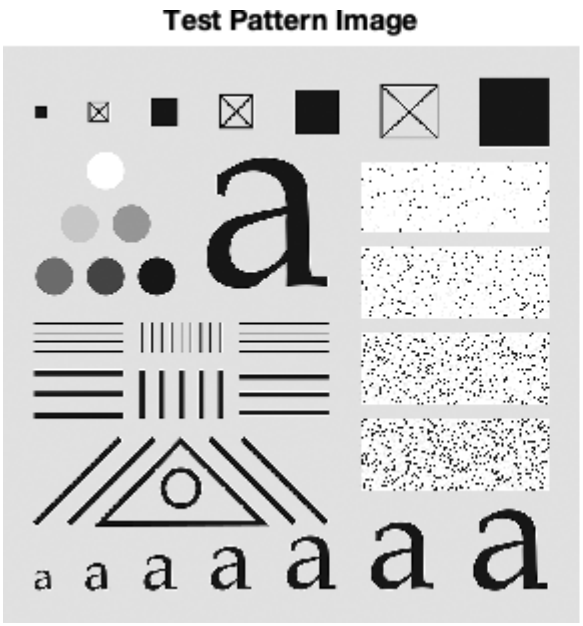


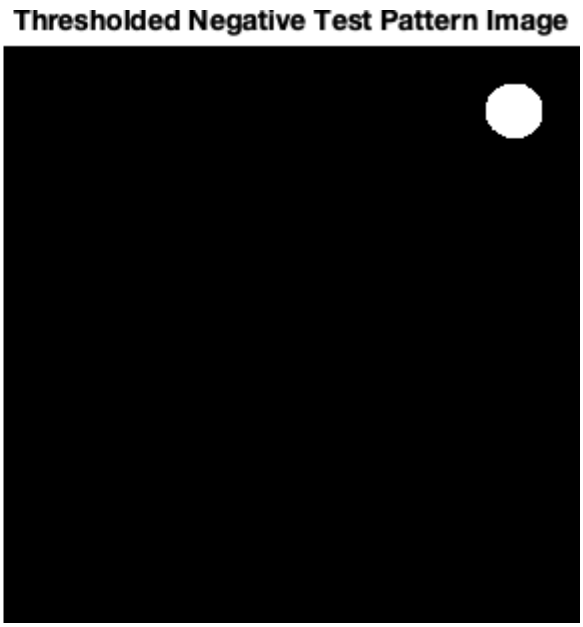

```
% Butterworth filter (parameters determined experimentally)
```

```
% Idea is to produce a severely blurred image
H = lpFilterTF4e('butterworth',P,Q,[8,2]);
g = dftFiltering4e(fn,H); % filtering in the frequency domain
figure('units','inches','Position',[0,0,8.25,4.25]);
subplot('position',[0,0,4/8.25,4/4.25]); imshow(fn);
title('Negative Test Pattern Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,4/4.25]); imshow(g);
title('Butterworth LP Filtered Image','FontSize',14);
```




```
% Threshold the image at a high value, say 80% of the max value
% which is determined experimentally
gT = g > 0.8*max(g(:));
figure('units','inches','Position',[0,0,8.25,4.25]);
subplot('position',[0,0,4/8.25,4/4.25]); imshow(f);
title('Test Pattern Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,4/4.25]); imshow(gT);
title('Thresholded Negative Test Pattern Image','FontSize',14);
```



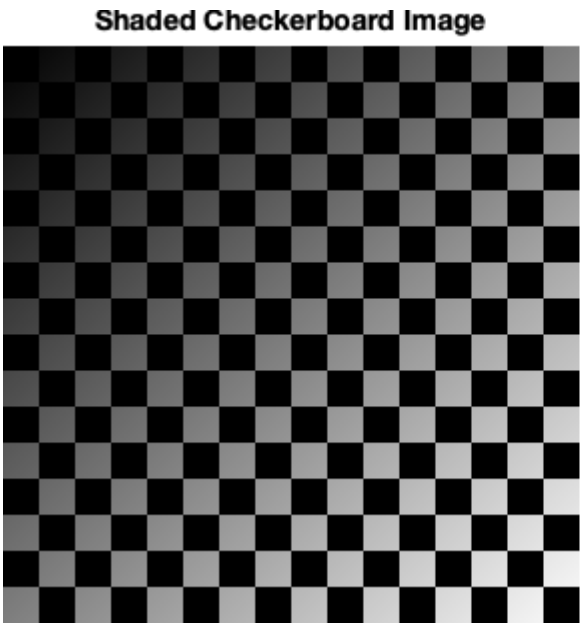


(c) Reproduce Results from Example 3.18

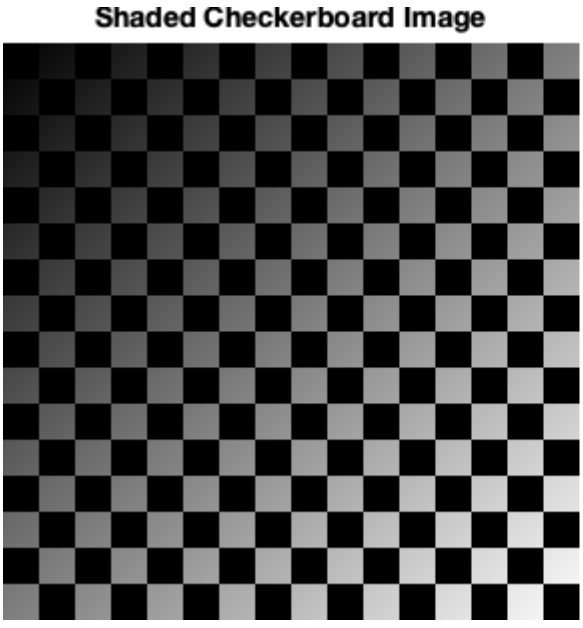
Read the image **`checkerboard1024-shaded.tif`** and reproduce the results in Example 3.18 using frequency-domain filtering.

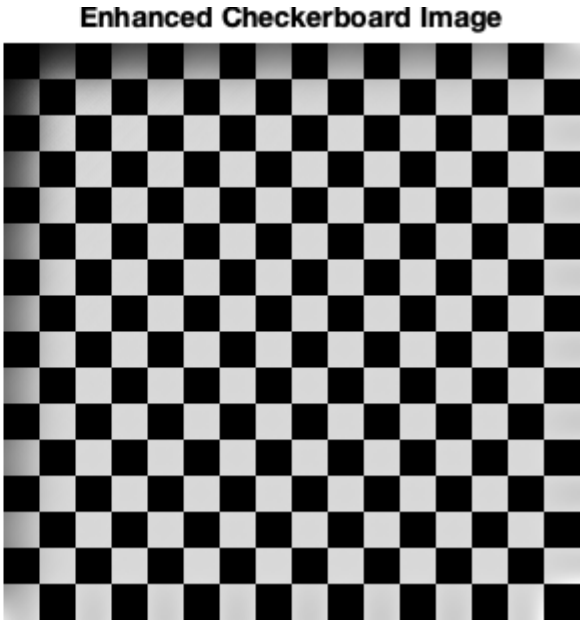
Solution: MATLAB script.

```
f = imread('../artfiles/checkerboard1024-shaded.tif');
% Since there will be division operation later, f has to be floating-point
f = intScaling4e(f);
[M,N] = size(f);
P = 2*M; Q = 2*N; % Paddig parameters
% The idea is to blur the image enough so that the squares are completely
% smoothed out, leaving only the shading pattern. This requires an
% aggresive bulrring filter
H = lpFilterTF4e('gaussian',P,Q,7); % determined experimentally
g = dftFiltering4e(f,H); % filtering in the frequency domain
figure('units','inches','Position',[0,0,8.25,4.25]);
subplot('position',[0,0,4/8.25,4/4.25]); imshow(f);
title('Shaded Checkerboard Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,4/4.25]); imshow(g);
title('Gaussian LP Filtered Image','FontSize',14);
```




```
% The corrected image is obtained by dividing the original shaded image
% by the shading pattern.
fclean = f./g;
% scale the corrected image so its intenities spans
% the full intensity scale
fclean = intScaling4e(fclean,'full');
figure('units','inches','Position',[0,0,8.25,4.25]);
subplot('position',[0,0,4/8.25,4/4.25]); imshow(f);
title('Shaded Checkerboard Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,4/4.25]); imshow(fclean);
title('Enhanced Checkerboard Image','FontSize',14);
```





Problem-5: DIP4E Project 4.9, parts (b) and (d) through (h) (Page 363)

Bandreject and bandpass filtering

(b) Bandpass Filter Function

Use the results from (a) to write a bandpass function, **`H = bpFilterTF4e(type,M,N,C0,W,n)`**, where the parameters are explained in (a).

Solution: MATLAB function.

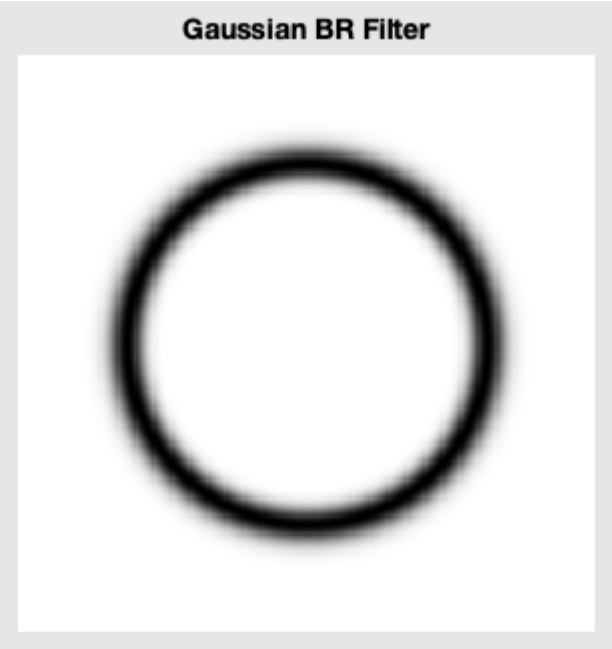
```
function H-bpFilterTF4e
%function H = bpFilterTF4e(type,M,N,C0,W,n)
%   H = BPFILTERTF4E(TYPE,M,N,CO,W,n) implements the three types of
%   bandpass filter transfer functions corresponding to the
%   bandreject functions in Table 4.7 of DIP4E. The bandpass
%   functions are obtained from the bandreject functions using Eq.
%   (4-148).
%   As explained in the text, when working with band filters we
%   normally use unpadded images to simplify interpreting frequency
%   components associated spatial features.

%   The specifications for qeuermm; the bnndreject mm mnsrer
%   Funcdons are as fo11cvns:

%   TYPE           PARAM         REIMRKS
%   'ideal'        C0            See Table 4.7 in DIP4E.
%   'gaussian'     C0            see Table 4.7 in DIP4E.
%   'butterworth' [C0, n]        see Table 4.7 in DIP4E.

% Copyrihht 2017. R. C. Gonzalez & R. E. Woods

%-Check number of input parameters.
if nargin == 5 % No value of n specified.
      n = 1; % Default value of n for Butterworth filter.
end

%-From Eq. (4.148) we know that a bandpass filter transfer function
% is obtained by subtracting the corresponding bandreject function
% from 1.
H = 1 - brFilterTF4e(type,M,N,C0,W,n);
```

(d) Gaussian Bandreject Filter.

Generate a 512×512 Gaussian bandreject filter transfer function using the same parameters as in (c) and show your result as an image.

Solution: MATLAB script.

```
clc; close all; clear;
HrGaussin = brFilterTF4e('gaussian',512,512,160,30);
figure('units','inches','Position',[0,0,4.25,4.5],'Color',0.9*[1,1,1]);
subplot('Position',[0.125/4.25,0.125/4.5,4/4.25,4/4.5]);
imshow(HrGaussin(1:2:end,1:2:end)); title('Gaussian BR Filter','FontSize',14);
```

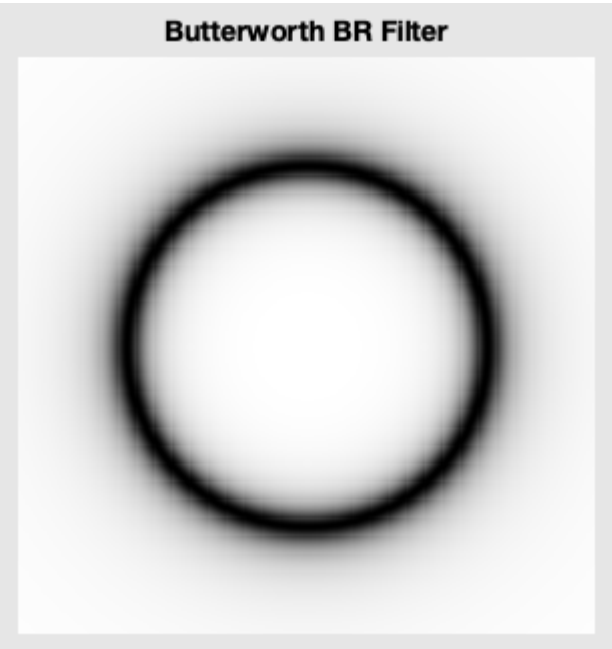


(e) Butterworth Bandreject Filter.

Generate a 512×512 Butterworth bandreject filter transfer function using the same parameters as in (c) and $n = 1$. Show your result as an image.

Solution: MATLAB script.

```
HrButterworth = brFilterTF4e('butterworth',512,512,160,30,1);
figure('units','inches','Position',[0,0,4.25,4.5],'Color',0.9*[1,1,1]);
subplot('Position',[0.125/4.25,0.125/4.5,4/4.25,4/4.5]);
imshow(HrButterworth(1:2:end,1:2:end)); title('Butterworth BR Filter','FontSize',14);
```

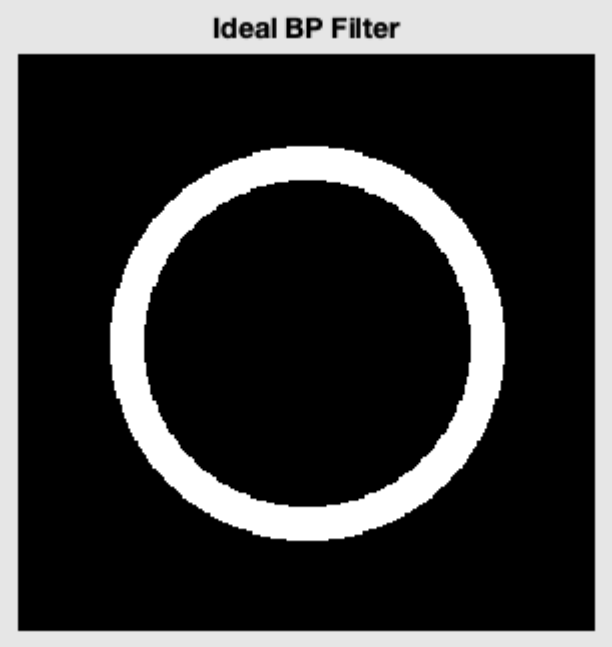


(f) Ideal Bandpass Filter.

Generate a 512×512 ideal bandpass filter transfer function using the same parameters as in (c) and show your result as an image.

Solution: MATLAB script.

```
HpIdeal = bpFilterTF4e('ideal',512,512,160,30);
figure('units','inches','Position',[0,0,4.25,4.5],'Color',0.9*[1,1,1]);
subplot('Position',[0.125/4.25,0.125/4.5,4/4.25,4/4.5]);
imshow(HpIdeal(1:2:end,1:2:end)); title('Ideal BP Filter','FontSize',14);
```

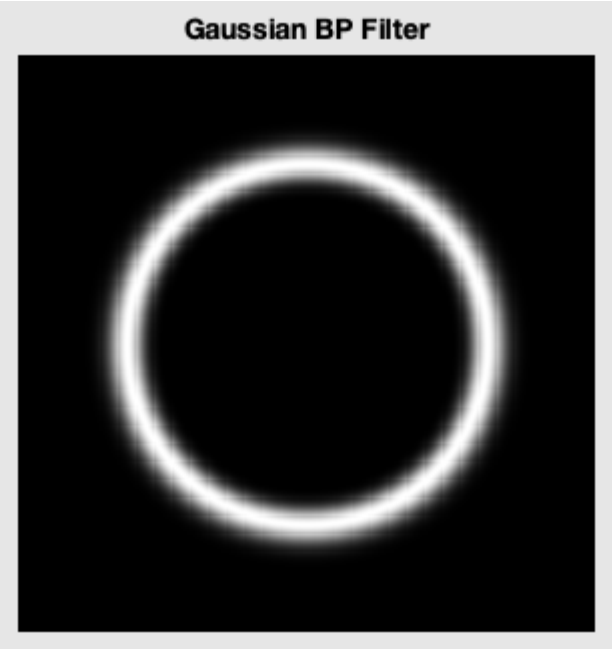


(g) Gaussian Bandpass Filter.

Generate a 512×512 Gaussian bandpass filter transfer function using the same parameters as in (c) and show your result as an image.

Solution: MATLAB script.

```
HpGaussian = bpFilterTF4e('gaussian',512,512,160,30);
figure('units','inches','Position',[0,0,4.25,4.5],'Color',0.9*[1,1,1]);
subplot('Position',[0.125/4.25,0.125/4.5,4/4.25,4/4.5]);
imshow(HpGaussian(1:2:end,1:2:end)); title('Gaussian BP Filter','FontSize',14);
```

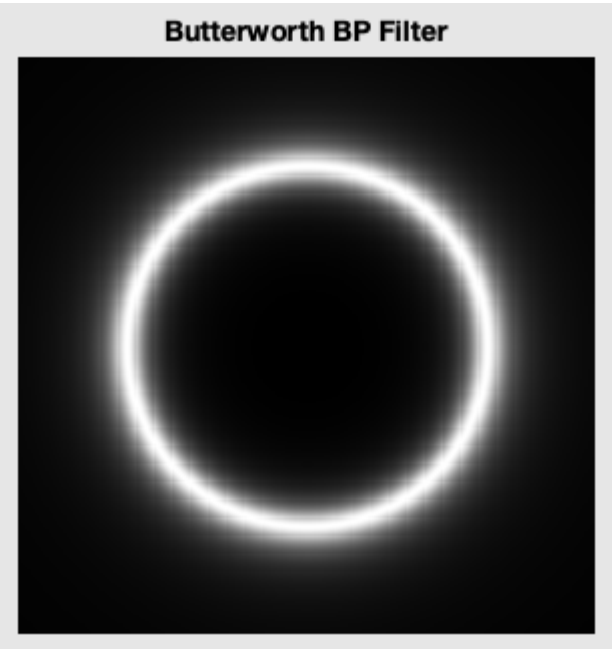


(h) Butterworth Bandpass Filter.

Generate a 512×512 Butterworth bandpass filter transfer function using the same parameters as in (c) and $n = 1$. Show your result as an image.

Solution: MATLAB script.

```
HpButterworth = bpFilterTF4e('butterworth',512,512,160,30,1);
figure('units','inches','Position',[0,0,4.25,4.5],'Color',0.9*[1,1,1]);
subplot('Position',[0.125/4.25,0.125/4.5,4/4.25,4/4.5]);
imshow(HpButterworth(1:2:end,1:2:end)); title('Butterworth BP Filter','FontSize',14);
```



Problem-6: DIPUM3E Project 4.1 (Page 241)

Working with phase information in images.

(a) Fourier Spectrum and Plots of `girl.tif` image

Read the image **`girl.tif`** and compute its spectrum and phase angle, as well as the real and imaginary parts of its DFT. Show the real and imaginary parts as images and note that they carry no meaningful visual information. Reconstruc the input image using on the spectrum information and repeat using the phase information. Display your results and explain why the two images look as they do.

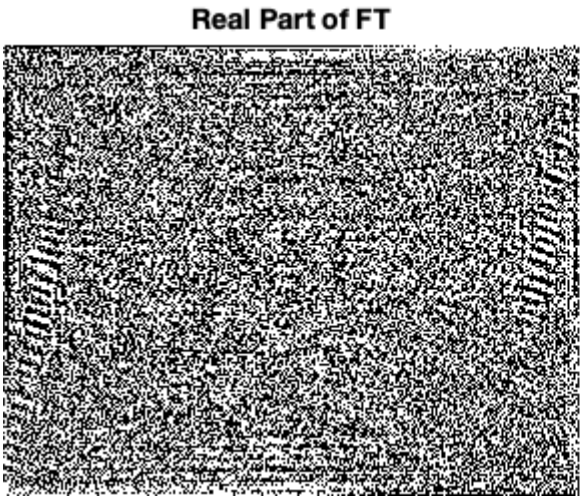
Solution:

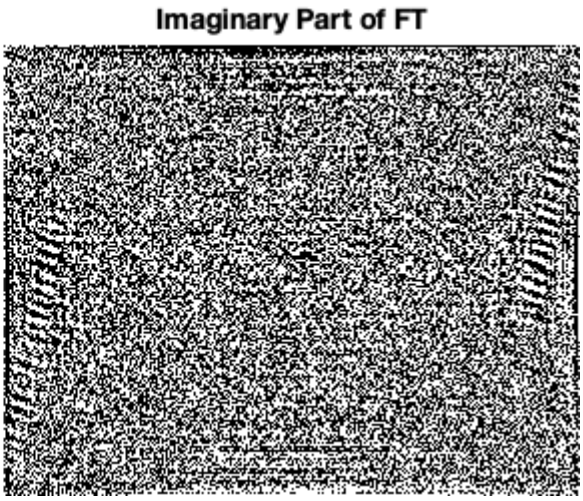
`girl.tif` image.

```
clc; close all; clear;
f = imread('../artfiles/girl.tif'); [M,N] = size(f);
figure('units','inches','Position',[0,0,4,3.5]);
subplot('Position',[0,0,1,M/N*4/3.5]); imshow(f);
title('Girl Image','fontsize',14);
```

Images of real and imaginary parts of the Fourier transform.

```
F = fft2(f); Freal = real(F); Fimag = imag(F);
Fmag = abs(F); Fpha = angle(F);
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]); imshow(Freal);
title('Real Part of FT','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(Fimag);
title('Imaginary Part of FT','FontSize',14);
```

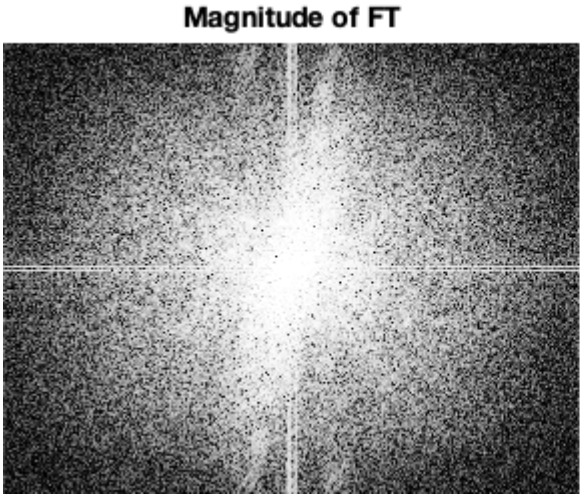


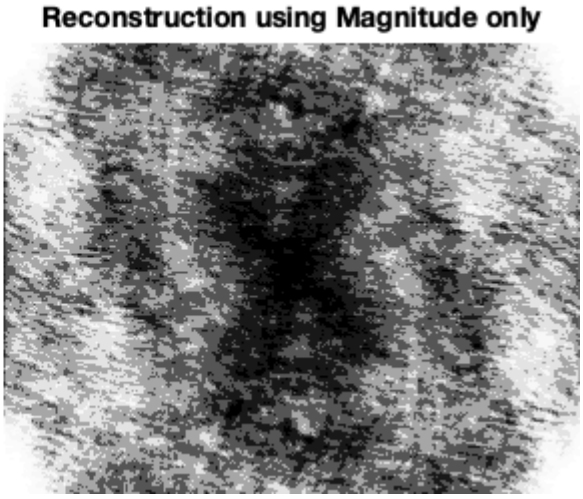


From the above images we see that the real and imaginary parts of the Fourier transform carries no visual information about the **`girl`** image.

Reconstruction using magnitude information only.

```
FmagShift = fftshift(Fmag); % center FT magnitude values
FmagShift = log10(FmagShift+1); % reduce dynamic range for display
FmagShift = mat2gray(FmagShift); % convert to intenisty image
FmagShift = histeq(FmagShift); % enhance for display
fm = real(ifft2(Fmag)); % reconstruct using magnitude only
fm = histeq(intensityScaling(fm)); % enhance reconstruction for display
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]); imshow(FmagShift);
title('Magnitude of FT','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(fm);
title('Reconstruction using Magnitude only','FontSize',14);
```



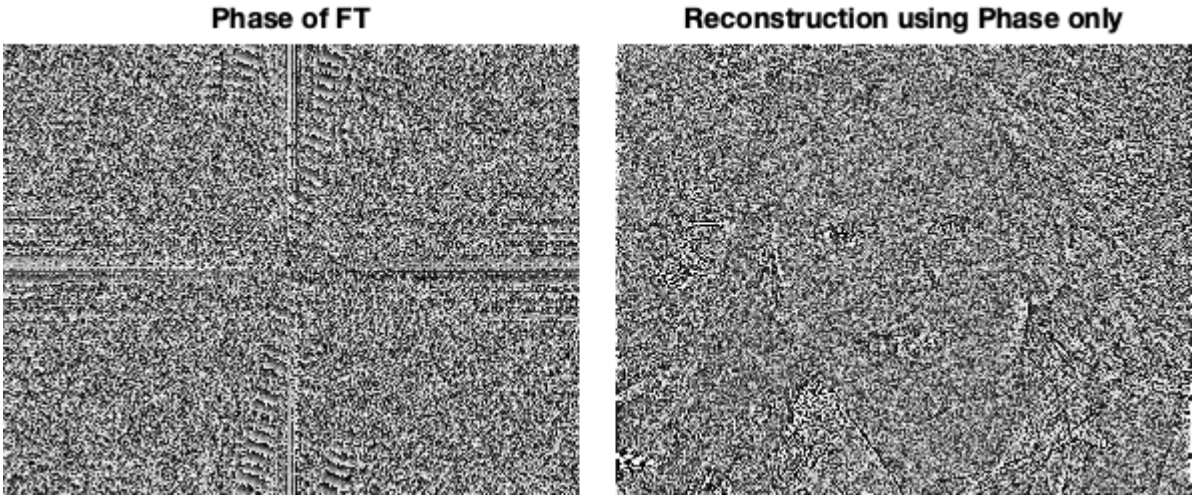


Clearly, there is no structure in the image on the right, only rough intensity variation.

Reconstruction using phase information only.

```
FphaShift = fftshift(Fpha); % center FT magnitude values
```

```
FphaShift = intensityScaling(FphaShift);
FphaShift = histeq(FphaShift); % enhance for display
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]);  imshow(FphaShift);
title('Phase of FT','FontSize',14);
Fp = 1*exp(complex(0,Fpha));
fp = real(ifft2(Fp)); %reconstruction using phase
fp = mat2gray(fp);
fp = histeq((fp)); % enhancement for display
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(fp,[]);
title('Reconstruction using Phase only','FontSize',14);
```



Note on image display: To save space, I displayed images with a width of 4 inches to make them smaller so I could fit two images in one row. However, as the above reconstructed image shows, a smaller width of 4 inch reconstruction (which is done using subsampling by the display driver) created so much aliasing that the facial details are lost. This clearly demonstrates the destructive effect of aliasing.

The same reconstruction is shown below using a width of 8 inches.

```
figure('units','inches','Position',[0,0,8,6.5]);
subplot('position',[0,0,1,M/N*8/6.5]); imshow(fp,[]);
title('Reconstruction using Phase only','FontSize',14);
```

The above reconstructed image, although has very poor constrast and intenisty tonality, clearly shows all visual details of facial image structure, confirming that phase carries visual information on image structure.

(b) Image Reconstruction using Negative Phase

Use the results from (a) to reconstruct the image using the spectrum and the negative of the phase. Display and explain the result.

Solution: MATLAB script.

```
Fb = Fmag.*exp(complex(0,-Fpha)); % reconstruction using negative phase angles
fb = real(ifft2(Fb));
fb = intensityScaling(fb);
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]); imshow(f);
title('Girl Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(fb);
title('Negative Phase Reconstruction','FontSize',14);
```

The reconstructed image is a rotated original girl image, which is to be expected because multiplication of the angle array by -1 is equivalent to rotating all elements of the array by 180°.

(c) Reconstruction using Phase of another Image.

Read the image girl.tif and letterA.tif. Compute the spectra and phase angles and do the following: (1) Generate an image by using the spectra of the letter image and phase angle of the girl image. (2) Form an image from the spectrum of the girl image and phase angle of the letter. Discuss the reasons for appearance of your results.

Solution: MATLAB script.

```
g = imread('../artfiles/letterA.tif'); SizeG = size(g), SizeF = size(f)
```

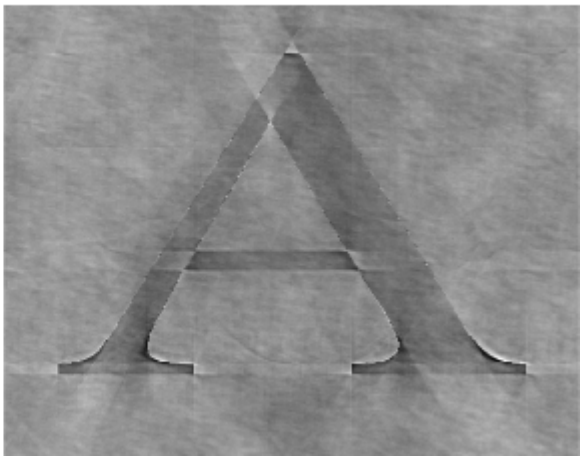
```
SizeG = 1×2
   450   450
SizeF = 1×2
   469   600
```

Since the size of the letter image is smaller than that of the girl image, we will have to pad the letter image by zeros on the right and bottom to make of size equal to that of the girl image. This can be done using the **`imresize`** function.

```
g = imresize(g,[size(f,1),size(f,2)]);
G = fft2(g); Gmag = abs(G); Gpha = angle(G);
% (1) Reconstruction using the spectra of the letter image and phase
% angle of the girl image.
Fc1 = Gmag.*exp(complex(0,Fpha)); fc1 = real(ifft2(Fc1));
fc1 = intensityScaling(fc1);
% (2) Reconstruction using the spectra of the girl image and phase
% angle of the letter image.
Fc2 = Fmag.*exp(complex(0,Gpha)); fc2 = real(ifft2(Fc2));
fc2 = intensityScaling(fc2);
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]); imshow(fc1);
title('Letter Spectra with Girl Phase','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(fc2);
title('Girl Spectra with Letter Phase','FontSize',14);
```

Letter Spectra with Girl Phase

Girl Spectra with Letter Phase



Discussion: The image on the left above shows that the girl dominates the visual structure of the result because the girl phase angles were used that carry the girl image information. However, we do see some visual contribution from letter image because its spectra is less varied or more concentrated. The image on the right shows even more clearly the domination of phase angle. The spectra of the girl is more varied or distributed than the original background of the letter A image. Thus, girl features are not observed.

Problem-7: DIPUM3E 4.5 (Page 243)

High-frequency emphasis implementation.

We know from Eq. (4-21) that high-frequency emphasis is implemented using a transfer function of the form:

$$H_{\mathrm{HFE}}(u,v) = a + bH_{\mathrm{HP}}(u,v)$$

where $H_{\mathrm{HP}}(u,v)$ is the transfer function of a highpass filter.

(a) `highfrequencyEmphasis` function

Write a function, **`[g,H] = highFrequencyEmphasis(f,a,b,D0,n)`**, that implements this filtering technique using a Butterworth highpass filter. Parameters **`a`** and **`b`** are as shown in the preceding equation, and **`D0`** and **`n`** are the parameters that define the Butterworth highpass filter transfer function. Use '**`symmetric`**' padding.

Solution: MATLAB function.

```
function [g,H] = highFrequencyEmphasis(f,D0,n,a,b)
% HIGHFREQUENCYEMPHASI5 High frequency emphasis fiitering.
%     [G,H] = HIGHFREQUENCYEMPHASIS(F,D0,N,A,B) implements high frequency
%     emphasis fiitering on input image F using a Butterworth highpass
%     filter having cutoff frequency D0 and order N. Parameters A and B are
%     as described in Eq. (4-21) in DIPUM3E. N defaults to 2, and A and B
%     default to 1. If A and B are provided, N must be provided aiso. In
%     the output, G is the processed image and H is the transfer used for
%     high frequency emphasis fiitering. Fiitering is done using padding,
%     so the filter transfer function is twice the size of F, and 2*DO
%     is used for the cutoff.
```

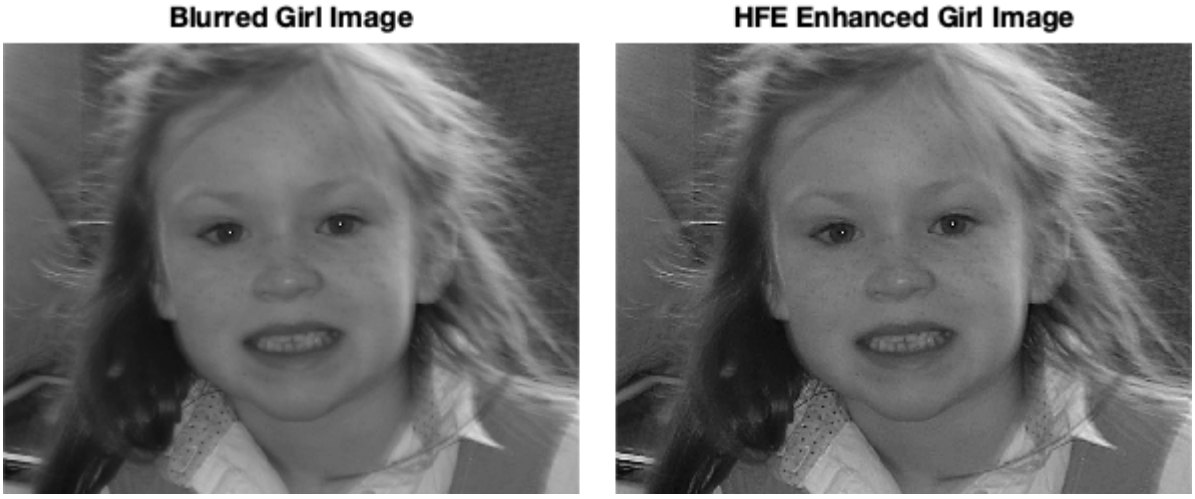
```
% Defaults.
if nargin == 2
   n = 2;
   a = 1;
   b = 1;
elseif nargin == 3
   a = 1;
   b = 1;
elseif nargin ~= 5
   error('wrong number of inputs')
end
% Image size.
[M,N] = size(f);
% Butterworth highpass transfer function.
BHP = hpfilter('butterworth',2*M,2*N,2*D0,n);
% High frequency emphasis fi1ter transfer function, from Eq. (4-21).
H = a + b*BHP;
% Do the fi1tering.
g = dftfilt(f,H,'symmetric');
end
```

(b) Sharpening of `girl-blurred.tif` Image

Read the image **`girl-blurred.tif`** and sharpen it using your function.

Solution: MATLAB script:

```
clc; close all; clear;
f = imread('../artfiles/girl-blurred.tif');
% First we will try default values, with the exception that D0 = 10% of the
% maximum image size
[M,N] = size(f); D0 = 0.1*max(M,N);
[g,~] = highFrequencyEmphasis(f,D0);
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]); imshow(f);
title('Blurred Girl Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(g);
title('HFE Enhanced Girl Image','FontSize',14);
```



The image on the right is slightly improved. But we can do better. Since the tonality is good, the value of a = 1 is fine. We will try a much higher value of b to increase the contribution of the filter.

```
[g,~] = highFrequencyEmphasis(f,D0,2,1,10);
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]); imshow(f);
title('Blurred Girl Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(g);
title('HFE Enhanced Girl Image','FontSize',14);
```

The enhanced image on the right shows that the value of b was too high. The image is oversharpened to the point of being distorted. let us try b = 6 instead.

```
[g,H] = highFrequencyEmphasis(f,D0,2,1,6);
figure('units','inches','Position',[0,0,8.25,3.5]);
subplot('position',[0,0,4/8.25,M/N*4/3.5]); imshow(f);
title('Blurred Girl Image','FontSize',14);
subplot('position',[4.25/8.25,0,4/8.25,M/N*4/3.5]); imshow(g);
title('HFE Enhanced Girl Image','FontSize',14);
```

This value of b worked perfectly. The image is actually a little shrper than the one given in the problem statement.

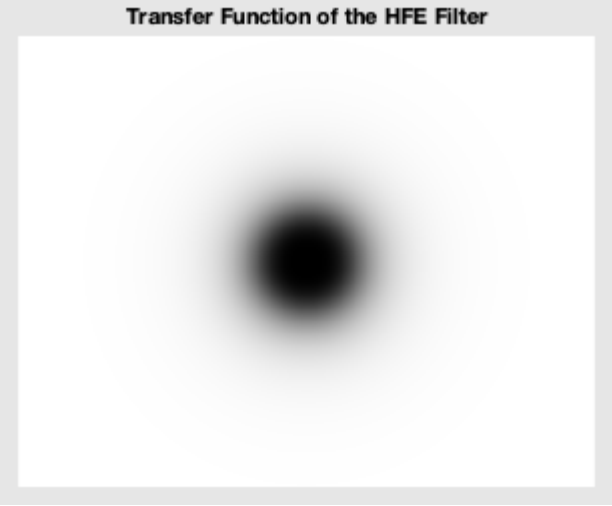
Note: The value od **`D0`** is not unique. For example, **`D0 = 5`** will also work.

(c) Display of HFE Transfer Function

Provide a display of the resulting HFE transfer function as an image. Show only one of four pixels in each dimension.

Solution: To understand the resulting high-frequency filter, let us display its transfer function as an image.

```
[MH,NH] = size(H);
figure('Units',"inches","Position",[0,0,4.25,3.5],'Color',0.9*[1,1,1]);
subplot('Position',[0.125/4.25,0.125/3.5,4/4.25,MH/NH*4/3.5]);
imshow(intensityScaling(fftshift(H)));
title('Transfer Function of the HFE Filter');
```



The above image clearly shows that the filter is a highpass filter.

Problem-8 DIPUM3E 4.7 (Page 244)

Sinusoidal Pattern

The Fourier transform of the 2-D sine function $f(x, y) = A \sin(2\pi u_0 x/M + 2\pi v_0 y/N)$ is

$$F(u, v) = j\frac{AMN}{2}\left(\delta(u + u_0, v + v_0) - \delta(u - u_0, v - v_0)\right),$$

where $\delta(u + u_0, v + v_0)$ denotes a unit impulse located at coordinates $(-u_o, -v_0)$ with respect to the center of the $M \times N$ frequency rectangle. The other impulse is the conjugate pair of the first, located at (u_0, v_0).

(a) `impulse2sin` function

Solution: MATLAB function.

```
function [r,S] = impulses2sin(M,N,C,A)
%IMPULSES2SIN Generates a 2-D spatia1 sinusoidal pattern from impulses.

%  [R,S] = IMPULSES2SIN(M,N,C,A), generates a spatiai sinusoidal
%      pattern, R, of size M—by-N, and its spectrum, S, in the frequency
%      domain. The input parameters are as fo11ows:
%
%      C is a K—by—2 matrix with K pairs of frequency domain coordinates
%      (u,v) that define the desired 1ocations of the impulses in the
%      frequency domain. The locations are with respect to the standard
%      origin, at the top 1eft of the rectang1e. On1y one pair of (u,v) =
%      (r,c) coordinates is specified for each impu1se. The program
%      computes the location of each conjugate. A11 impu1ses must be
%      located in the M—by-N frequency rectang1e.
%
%  Because we work with center transforms, the dc term is at
%  coordinates [f1oor(M/2) + 1, f1oor(N/2) + 1]. For examp1e, if M = N
%  = 600, the cennter is at (301,301). If you want to specify an
%  impuise 2 cyc1es away from dc whose corresponding sine wave is
%  oriented in the +45 deg counterc1ockwise direction with respect to
%  the verticai axis, you cou1d specify C = [299,299], which is in the
%  upper 1eft quadrant with respect to the center. Or, equivaientiy,
%  you can specify its conjugate at C = [303,303] in the 1ower right
%  quadrant.

%      A is a 1—by—K vector that contains the amplitude of each of the K
%      impulse pairs. If A is not inc1uded in the argument, the defau1t
%      used is A = oNEs(1,K).

% Number of impu1ses.
K = size(C,1);
% Defaults.
if nargin < 4
    A = ones(1,K);
end
% All coordinates in C must be inside the frequency rectangle.
if any(C(:,1) > M) || any(C(:,2) > N)
    error('All impulses must be inside the frequency rectangle')
end
% center of the frequency rectangle.
r0 = floor(M/2) + 1; c0 = floor(N/2) + 1;
% Distance differences from the specified frequencies to the center of
% the frequency rectangle.
delr(:) = r0 - C(:,1); delc(:) = c0 - C(:,2);
% coordinates of complex conjugates.
CC(:,1) = C(:,1) + 2*delr(:); CC(:,2) = C(:,2) + 2*delc(:);
% Insert coodinates of the impu1ses and their comp1ex conjugates into a
% Fourier transform of all Os;
F = zeros(M,N);
for k = 1:K
    F(C(k,1),C(k,2)) = 1j*M*N*(A(k)/2);
    F(CC(k,1),CC(k,2)) = -1j*M*N*(A(k)/2);
end
S = abs(F);
% work with the real part in case of parasitic complex components in the
% inverse.
r = real(ifft2(ifftshift(F)));
```

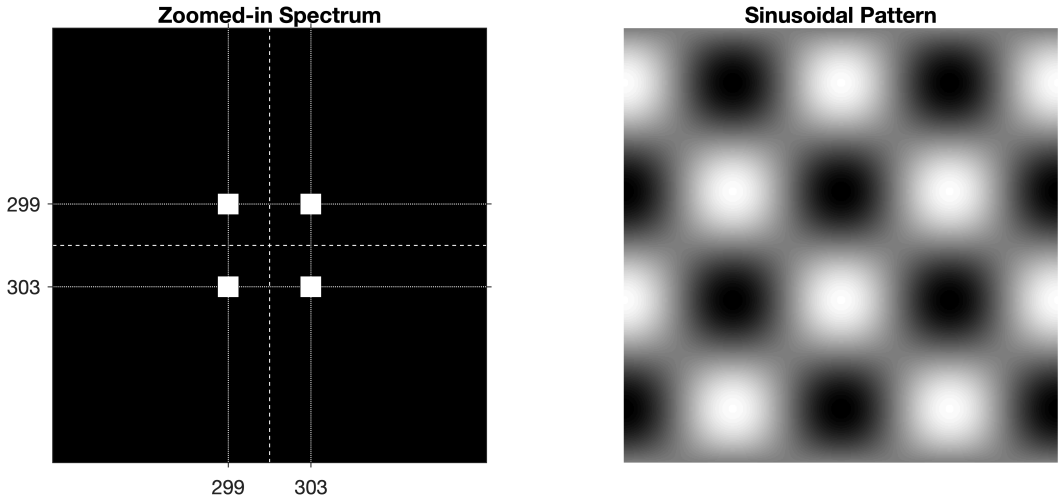
```
end
```

(b) Generation of Sinusoidal Pattern using `impulse2sin` Function

The 600×600 sinusoidal pattern shown on the left image in the problem statement was generated by using **`impulses2sin`** function. Generate and display a pattern that looks like it.

Solution: From the left figure in appears that the main directions of the wavefronts are $+45^{\circ}$ and -45° and are of very low frequencies. After few trials we have the following solution.

```
clc; close all; clear;
M = 600; N = 600;
C = [299,299;299,303];
[r,S] = impulses2sin(M,N,C); r = intensityScaling(r);
figure('Units',"inches","Position",[0,0,9,4]);
subplot(1,2,1); imshow(S(291:311,291:311)); title('Zoomed-in Spectrum');
hold on; plot([11,11],[0,22],'w--'); plot([0,22],[11,11],'w--');
plot([9,9],[0,22],'w:'); plot([0,22],[9,9],'w:');
plot([13,13],[0,22],'w:'); plot([0,22],[13,13],'w:'); hold off;
axis on; set(gca,'xtick',[9,13],'xticklabel',{'299','303'},...
    'ytick',[9,13],'yticklabel',{'299','303'});
print -dpng zoomedSpectrum.png;
subplot(1,2,2); imshow(r); title('Sinusoidal Pattern');
```



Note: To see impulse pixels in the spectum display, I have shown zoomed (exploded) spectrum display, from 291 through 311 pixels in each dimension. The spectrum values outside of this region are zero.

(c) Filtering of the Sinusoidal Pattern

Filter the pattern in (b) so that it looks like the image on the right in the problem statement.

Solution: From (b) we know that the impulses that generated the patterns are at (299,299) and (299,303). Their conjugates were generated automatically by the function. The given pattern is clearly a single sinusoid so we have to eliminate one of the conjugate pairs. The given pattern is oriented from the top left to the right bottom

quadrants, so the impulse we need to eliminate is the one at [299,303]. Since we know the exact location we can use an ideal notch fi'|ter.

```
% Eliminate the notch at (299,303). Since it is a sing1e pixel, the
% radius of the notch can be small, say, 3.
C1 = [299 303]; H = cnotch('ideal','reject',600,600,C1,3);
g = dftfilt(r,H); g = intensityScaling(g);
% The following resu1t agrees with the image shown in the project statement.
figure('Units',"inches","Position",[0,0,9,4]);
subplot(1,2,1); imshow(g); title('Desired Pattern');
```