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

Due on October 19, 2021 by 11:59 pm via submission portal

NAME: McKean, Tyler

Instructions

1. You are required to complete this assignment using Live Editor.
2. Enter your MATLAB script in the spaces provided. If it contains a plot, the plot will be displayed after the script.
3. All your plots must be properly labeled and should have appropriate titles to get full credit.
4. Use the equation editor to typeset mathematical material such as variables, equations, etc.
5. After completeing this assignment, export this Live script to PDF and submit the PDF file through the provided submission portal.
6. You will have two attempts to submit your assignment. However, make every effort to submit the correct and completed PDF file the first time. If you use the second attempt, then that submission will be graded.
7. Your submission of problem solutions must be in the given order, i.e., P1, P2, P3, etc. Do not submit in a random order.
8. Please submit your homework before the due date/time. A late submission after midnight of the due date will result in loss of points at a rate of 10% per hour until 8 am the following day, at which time the solutions will be published.

Reading Assignment: Chapters 4 from DIP4E and DIPUM3E.

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:

$f(m,n) = f_c(x,y)|_{x=m\Delta x, y=n\Delta y} = f_c(m\Delta x, n\Delta y)$, where $\xi_x = 2 = \frac{1}{\Delta x}, \xi_y = 3 = \frac{1}{\Delta y}$ then $\Delta x = \frac{1}{2}$ and $\Delta y = \frac{1}{3}$, plugging these values into $f_c(x, y)$ we get

$f(m,n) = 3cos\left(2.4\pi\left(\frac{1}{2}\right) + 2.6\pi\left(\frac{1}{3}\right)\right) = 3cos(1.2\pi m + 0.867\pi n)$ then to create the reconstructed signal we'd use the formula:

$\hat{f}_c(x,y) = \sum_m \sum_n f(m,n) sinc(\xi_x(x - m\Delta x)) sinc(\xi_y(y - n\Delta y))$ and plug in the value of $f(m, n)$ we'd get

$$\hat{f}_c(x,y) = \sum_m \sum_n 3cos(1.2\pi m + 0.867\pi n) sinc(2(x - \frac{m}{2})) sinc(3(y - \frac{n}{3}))$$

which would result in

$$\hat{f}_c(x,y) = 3cos(1.6\pi x + 2.6\pi y)$$

Here the sample frequency for the X component was less than twice the max frequency of 2.4, thus this signal component will contain aliasing at the 2nd harmonic. However, for the Y component, using the sample frequency equal to 3 satisfies this requirement and will no have any aliasing occur. Then the reconstructed signal we get contains the same frequency of the original y component but has a different frequency for the x component.

(b) Sampling Rate-2

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

Solution:

$f(m,n) = f_c(x,y)|_{x=m\Delta x, y=n\Delta y} = f_c(m\Delta x, n\Delta y)$, where $\xi_x = 3 = \frac{1}{\Delta x}, \xi_y = 2 = \frac{1}{\Delta y}$ then $\Delta x = \frac{1}{3}$ and $\Delta y = \frac{1}{2}$, plugging these values into $f_c(x, y)$ we get

$f(m,n) = 3cos\left(2.4\pi\left(\frac{1}{3}\right) + 2.6\pi\left(\frac{1}{2}\right)\right) = 3cos(0.8\pi m + 1.3\pi n)$ then to create the reconstructed signal we'd use the formula:

$\hat{f}_c(x,y) = \sum_m \sum_n f(m,n) sinc(\xi_x(x - m\Delta x)) sinc(\xi_y(y - n\Delta y))$ and plug in the value of $f(m,n)$ we'd get

$$\hat{f}_c(x,y) = \sum_m \sum_n 3cos(0.8\pi m + 1.3\pi n) sinc(3(x - \frac{m}{3})) sinc(2(y - \frac{n}{2}))$$

which would result in

$$\hat{f}_c(x,y) = 3cos(2.4\pi x + 1.4\pi y)$$

This situation is the opposite of the previous, where now the Y component will have aliasing occur in its 2nd harmonic and result in a lower frequency when reconstructing the signal, but the X component will have no aliasing occur and contain the same frequency as the original signal.

(c) Sampling Rate-3

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

Solution:

$f(m,n) = f_c(x,y)|_{x=m\Delta x, y=n\Delta y} = f_c(m\Delta x, n\Delta y)$, where $\xi_x = 3 = \frac{1}{\Delta x}, \xi_y = 3 = \frac{1}{\Delta y}$ then $\Delta x = \frac{1}{3}$ and $\Delta y = \frac{1}{3}$, plugging these values into $f_c(x, y)$ we get

$f(m,n) = 3cos\left(2.4\pi\left(\frac{1}{3}\right) + 2.6\pi\left(\frac{1}{3}\right)\right) = 3cos(0.8\pi m + 0.867\pi n)$ then to create the reconstructed signal we'd use the formula:

$\hat{f}_c(x,y) = \sum_m \sum_n f(m,n) sinc(\xi_x(x - m\Delta x)) sinc(\xi_y(y - n\Delta y))$ and plug in the value of $f(m,n)$ we'd get

$$\hat{f}_c(x,y) = \sum_m \sum_n 3cos(0.8\pi m + 0.867\pi n) sinc(3(x - \frac{m}{3})) sinc(3(y - \frac{n}{3}))$$

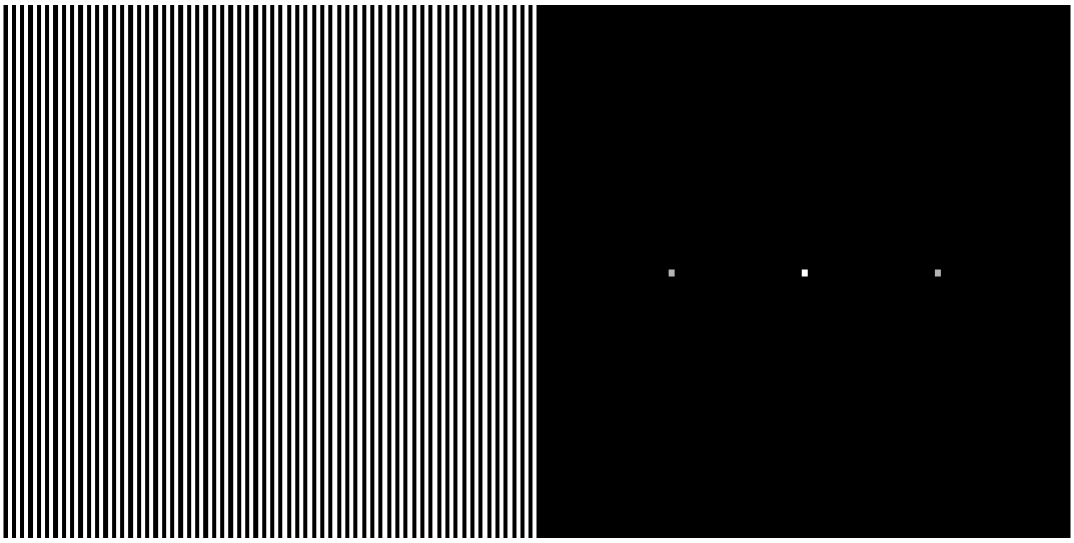
which would result in

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

Finally, both the sampling frequencies meet the criteria to have no aliasing occur because now both values of ξ are large enough such that no harmonics will be aliased. This results in the reconstructed signal being equivalent to the original signal that was sampled.

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.



(b) Why Horizontal Spectrum?

Solution:

The image on the left had a centered Fourier Transform performed on it where the dc component, $F(0,0)$, is now placed in the center of the spectrum image. The components of the spectrum are only displaying on the horizontal axis because the image on the left consists of an evenly symmetric sinusoidal signal in the N direction. This would correlate to a cosine function with its dc term centered in the spectrum and impulses spread out symmetrically along the horizontal axis shifted by the fundamental frequency of the cosine.

(c) New Spectrum

Solution:

If the image on the left consisted of stripes that are one pixel apart rather than two, the spectrum of the image would still contain a centered dc component but the impulses being shifted by the fundamental frequency within the cosine would now be twice as far apart because, in the spatial domain, the pixels reduced in length. This is due to the fact that a smaller spatial distances in the spatial domain correlates to larger frequencies in the frequency domain.

(d) DC Terms?

Solution:

The DC term would remain unchanged. Changing the spatial distance of the left image from two pixels per stripe to one pixel per stripe would only change the frequency shift of the impulses in the frequency domain. Thus, spectral image would still contain a centered dc component.

Problem-3: DIP4E Problem 4.47 (Page 359)

Solution:

Let the image on the left be $f(x, y)$ and the resulting image be known as $g(x, y)$

For part a.) of this process, the operation of $f(x, y)(-1)^{x+y}$ allows the spectrum to be centered about the image when taking the DFT of $f(x, y)$ to compensate for the periodicty property of the Fourier Transform.

Part b.) the centered input image $f(x, y)(-1)^{x+y}$, then undergoes the DFT to receive the sprectral information of the image, $F(u, v),$ where the spectrum is now centered about the intervals of $[0, M-1]$ and $[0, N-1]$ or $F\left(u-\frac{M}{2}, v-\frac{N}{2}\right)$

Part c.) the spectrum $F(u, v)$ is then expressed as the complex conjugate, so $F^*(u, v)$, which conducts a symmetry property between the spatial and frequency domains such that, $f(-x, -y) real \Longleftrightarrow F^*(u, v) complex$

So then in Part d.) when then reverse the image back to the Spatial Domain through the Inverse DFT, all the values of the image have been symmetrical flipped to their negative values.

Finally, the image is restored back to it's original dimensions by multiplying the real part of the result by $(-1)^{x+y}$

Which, through this entire process, the resulting image $g(x, y)$ is an upside-down version of $f(x, y)$.

Problem-4: DIP4E Project 4.6 (Page 363)

Lowpass filtering in the frequency domain

(a) Gaussian Lowpass Filtering

Solution:

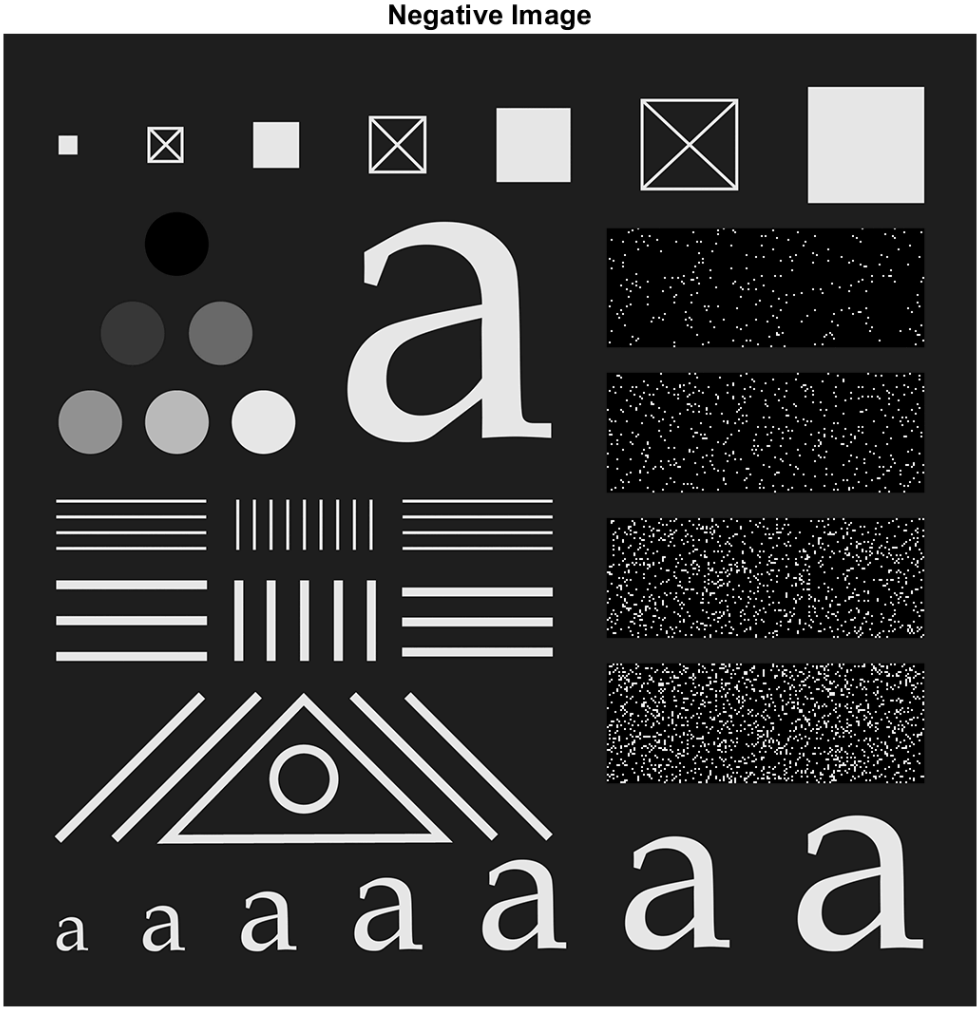
```
f = imread('testpattern1024.tif');
[f1, revertClass] = tofloat(f);
sigma = 6;
PQ = paddedsize(size(f));
Fp = fft2(f1,PQ(1),PQ(2)); % Compute the FFT with zero padding
Hp = lpfilter('gaussian',PQ(1),PQ(2),2*sigma);
Gp = Hp.*Fp;
gp = ifft2(Gp);
gpc = gp(1:size(f,1),1:size(f,2));
gpc = revertClass(gpc);
figure, imshow(gpc),title('LPF in Frequency Domain'),pause(1)
```

LPF in Frequency Domain

(b) Butterworth Lowpass Filtering and Thresholding

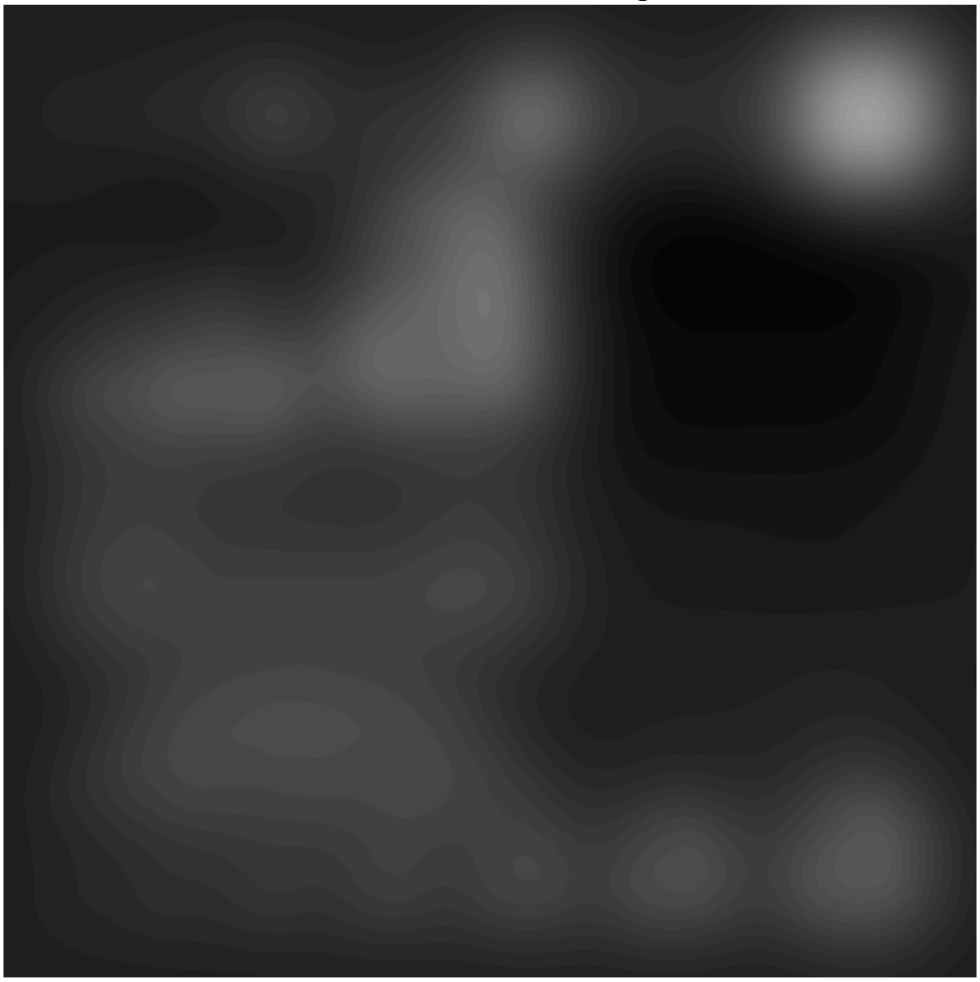
Solution:

```
f2 = intXform4e(f,'negative');
figure, imshow(f2),title('Negative Image'), pause(1)
```

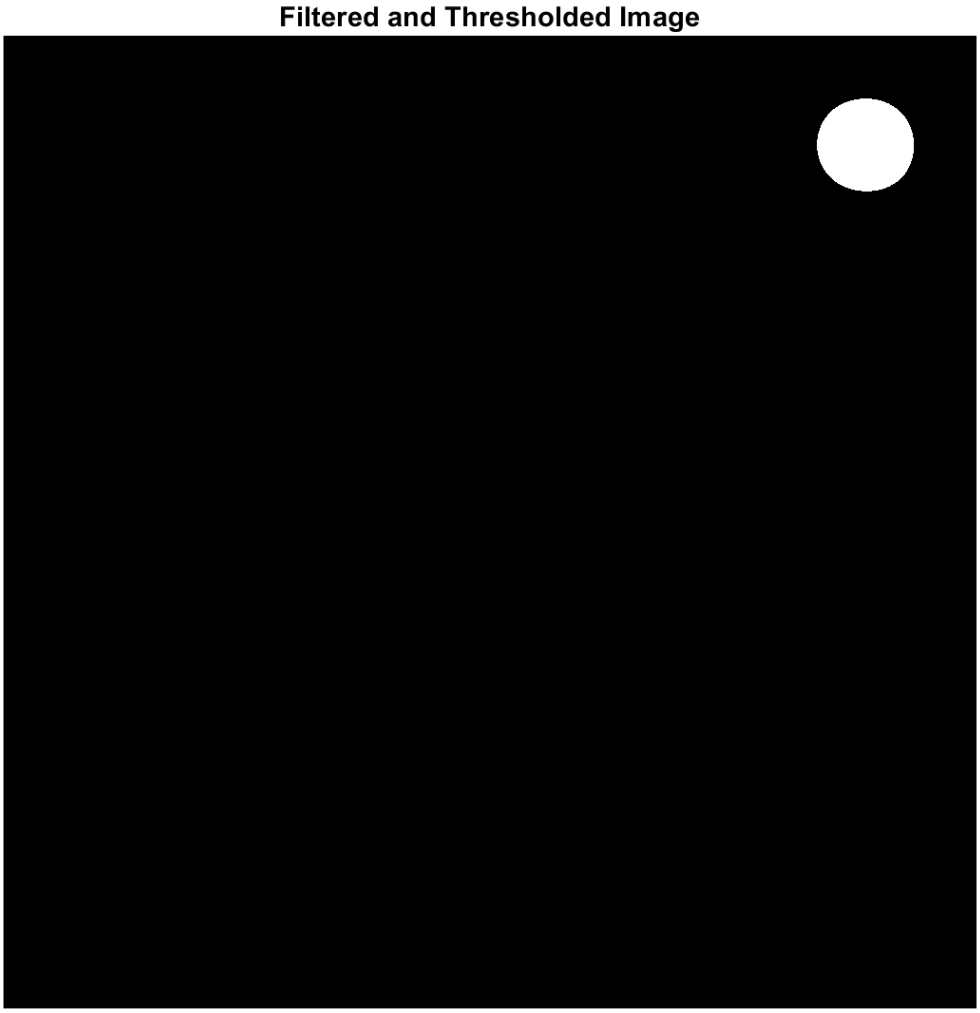



```
PQ = paddedsize(size(f));
h2 = lpFilterTF4e('butterworth',PQ(1),PQ(2),[8,2]);
g2 = dftFiltering4e(f2,h2);
figure, imshow(g2), title('Butterworth Filtered Image'), pause(1)
```

Butterworth Filtered Image



```
g_thresh = g2 > 0.8*max(g2(:));
figure, imshow(g_thresh), title('Filtered and Thresholded Image'), pause(1)
```

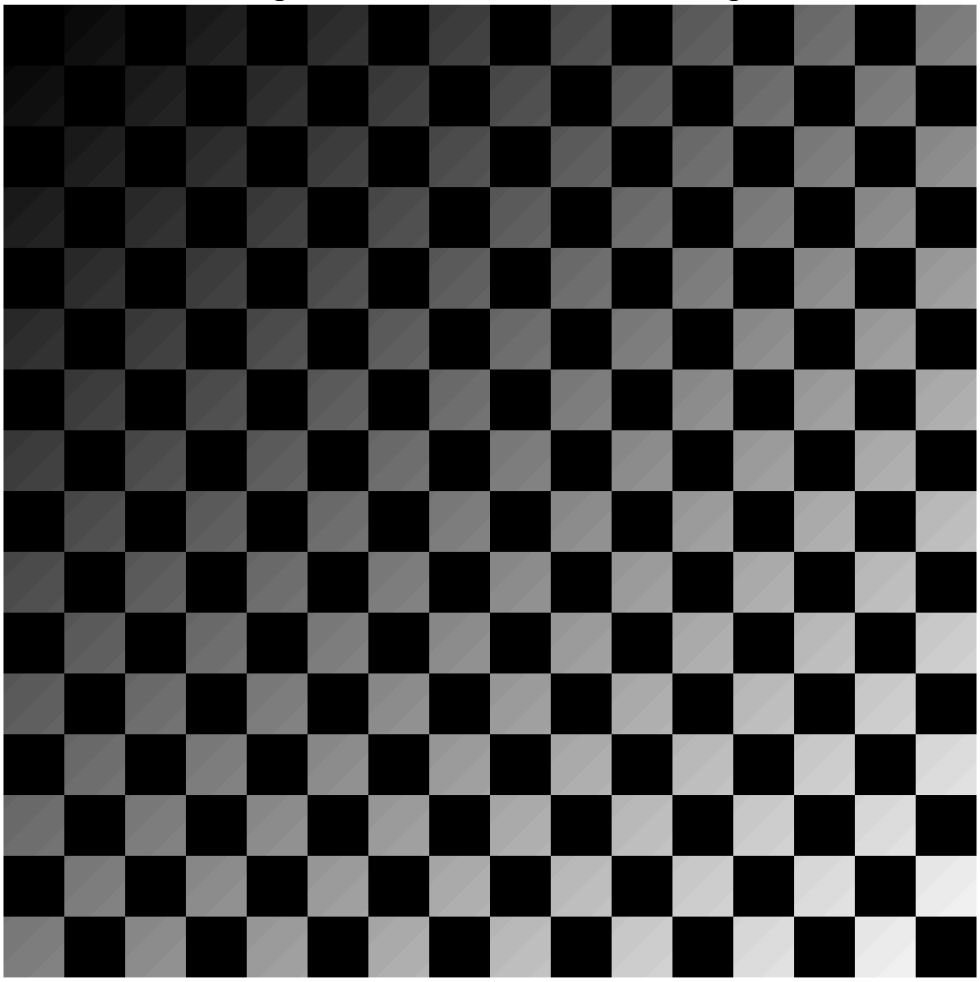


(c) Results from Example 3.18

Solution:

```
f3 = imread('checkerboard1024-shaded.tif');
f3 = intScaling4e(f3);
figure, imshow(f3), title('Integer Scaled Checkerboard1024 Image'),pause(1)
```

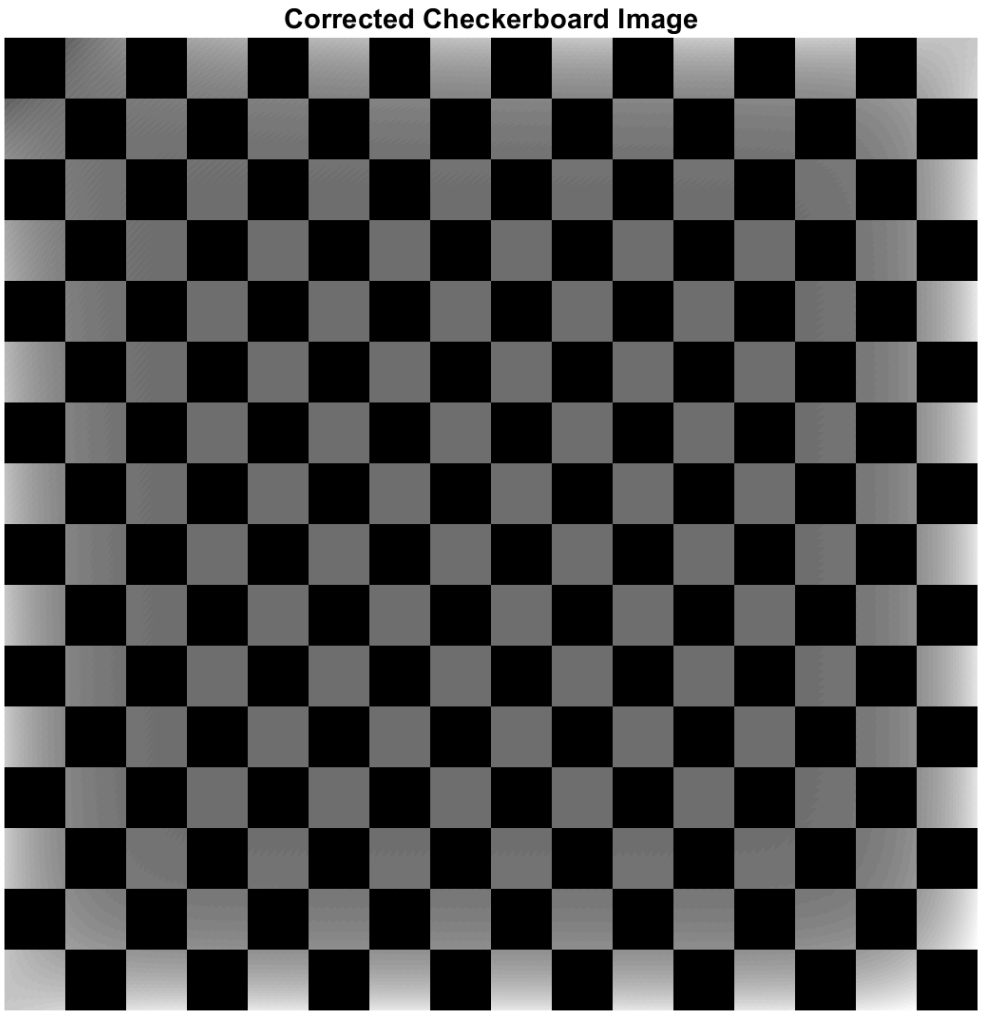
Integer Scaled Checkerboard1024 Image



```
[f3, revertClass] = tofloat(f3);
sigma = 2;
PQ = paddedsize(size(f3));
F3 = fft2(f3,PQ(1),PQ(2)); % Compute the FFT with zero padding
H3 = lpfilter('gaussian',PQ(1),PQ(2),2*sigma);
G3 = H3.*F3;
g3 = ifft2(real(G3));
shaded = g3(1:size(f3,1),1:size(f3,2));
shaded = revertClass(shaded);
shaded = intScaling4e(shaded);
figure, imshow(shaded),title('Shading Pattern from Filtering in Frequency Domain'),pause(1)
```

Shading Pattern from Filtering in Frequency Domain

```
fCorrected = f3./shaded;
fCorrected = intScaling4e(fCorrected);
figure, imshow(fCorrected), title('Corrected Checkerboard Image'),pause(1)
```



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

Bandreject and bandpass filtering

(b) Bandpass Filter Function

Solution: Insert the code of your function below after the comments.

```
function H = bpFilterTF4e(type,band,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.

% Insert your code below.

%Protect against uppercase
type = lower(type);
band = lower(band);
%Use function dftuv to set up meshgrid arrays needed for computing the required distances
[U,V] = dftuv(M,N);
%Compute the distances D(U,V)
D = hypot(U,V);
% Determine if need to use default n.
if nargin < 7
    n = 1; %Default BW filter order.
end
%Begin filter computations
switch type
  case 'ideal'
     RI = D <= C0 - (W/2);
     R0 = D >= C0 + (W/2);
     H = tofloat(R0|RI);
  case 'butterworth'
     H = 1./(1 + ((D*W)./(D.^2 - C0^2)).^(2*n));
  case 'gaussian'
     H = 1 - exp(-((D.^2 - C0^2)./(D.*W + eps)).^2);
  otherwise
     error('Unknown filter type.')
end

%Convert to a bandpass transfer function if specified
if isequal(band,'pass')
     H = 1 - H;
end
```

(d) Gaussian Bandreject Filter.

Solution:

```
M = 512;
N = 512;
C0 = 128;
W = 90;
n = 3;
H = bpFilterTF4e('gaussian','reject',M,N,C0,W,n);
Hc = fftshift(H);
figure, imshow(Hc),title('Gaussian Bandreject Filter'),pause(1)
```

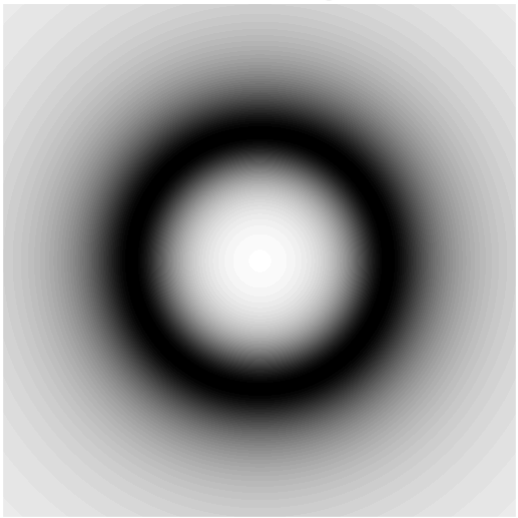
Gaussian Bandreject Filter

(e) Butterworth Bandreject Filter.

Solution:

```
n = 1;
H = bpFilterTF4e('butterworth','reject',M,N,C0,W,n);
Hc = fftshift(H);
figure, imshow(Hc),title('Butterworth Bandreject Filter'),pause(1)
```

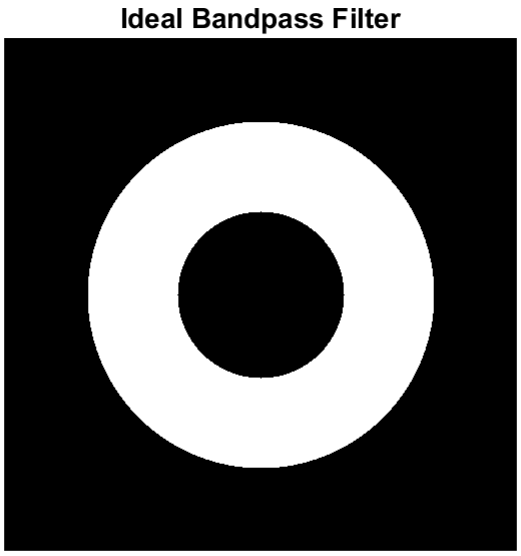
Butterworth Bandreject Filter



(f) Ideal Bandpass Filter.

Solution:

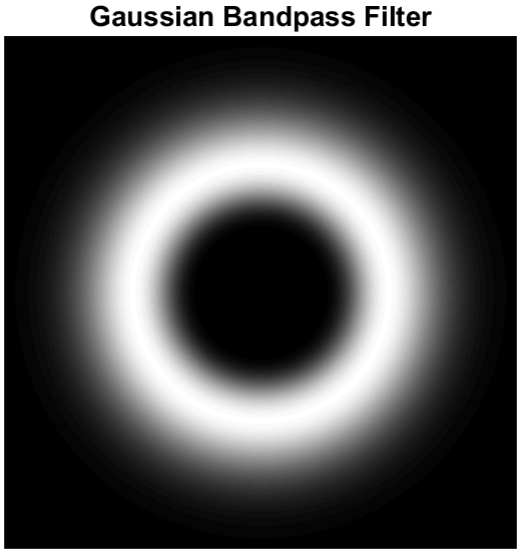
```
H = bpFilterTF4e('ideal','pass',M,N,C0,W,n);
Hc = fftshift(H);
figure,imshow(Hc),title('Ideal Bandpass Filter'),pause(1)
```



(g) Gaussian Bandpass Filter.

Solution:

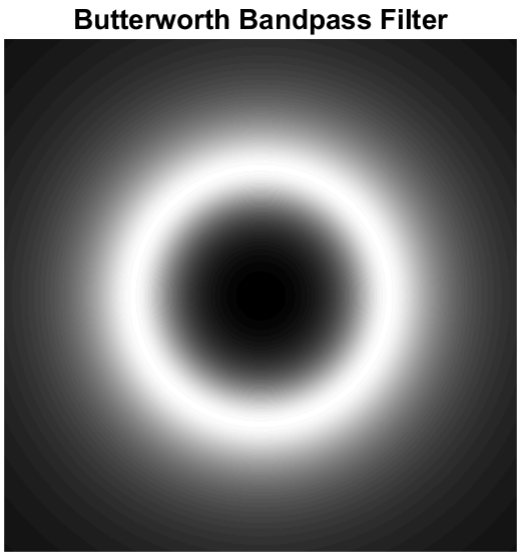
```
H = bpFilterTF4e('gaussian','pass',M,N,C0,W,n);
Hc = fftshift(H);
figure,imshow(Hc),title('Gaussian Bandpass Filter'),pause(1)
```



(h) Butterworth Bandpasst Filter.

Solution:

```
H = bpFilterTF4e('butterworth','pass',M,N,C0,W,n);
Hc = fftshift(H);
figure,imshow(Hc),title('Butterworth Bandpass Filter'),pause(1)
```



Problem-6: DIPUM3E Project 4.1 (Page 241)

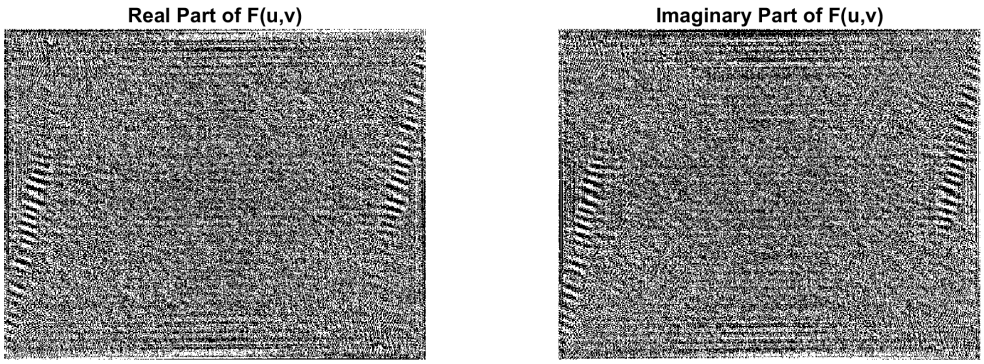
Working with phase information in images.

(a) Spectrum and Phase Angle of `girl.tif` image

Solution:

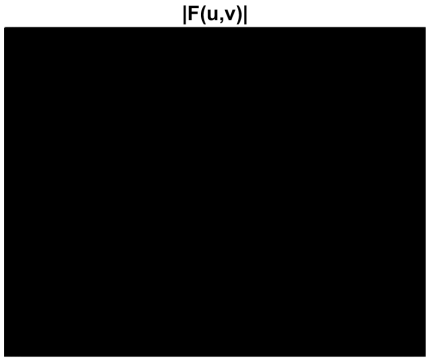
```
f_girl = imread('girl.tif');
figure, imshow(f_girl),title('Girl Image'),pause(1)
```

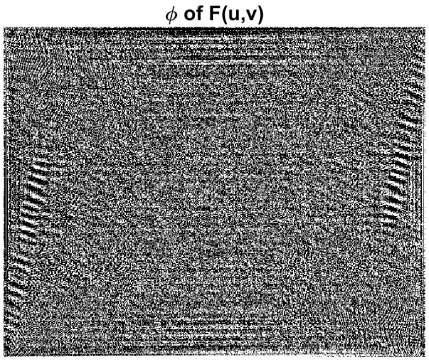

```
[f, revertClass] = tofloat(f_girl);
F = fft2(f);
F_real = real(F);
F_imag = imag(F);
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(F_real); title('Real Part of F(u,v)','fontsize',14);
subplot(1,2,2); imshow(F_imag); title('Imaginary Part of F(u,v)','fontsize',14);pause(1)
```



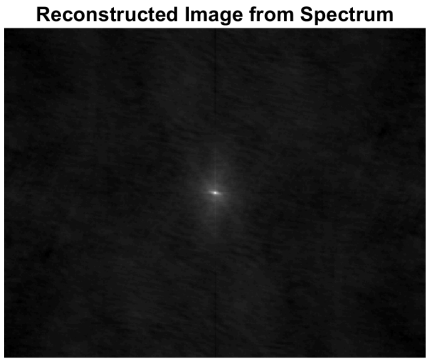
Both the real and imaginary parts of the image don't contain any useful information as displayed by the figures above

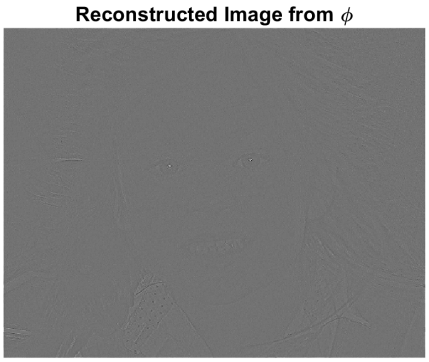
```
S = abs(F); % Magnitude of F(u,v)
phi = atan2(F_imag,F_real); % Phase of F(u,v)
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(S,[]); title('|F(u,v)|','fontsize',14);
subplot(1,2,2); imshow(phi); title('\phi of F(u,v)','fontsize',14);pause(1)
```






```
% Reconstruct the image using only the spectrum
S0 = complex(S,0);
% Center the result
FS = fftshift(real(ifft2(S0)));
% The spectrum has a large dynamic range, so use its log instead to
% compress the dynamic range. Also scale the intensities to the range [0 1]
% for display
FS = intensityScaling(log10(1 + FS));

% As following image shows, there's no structure to the result, only rough
% inensity variations. As mentioned in the project statement, this is the
% image content carried by the spectrum, so we would not expect to see any
% features resembling the input image.
% Now reconstruct the image using only the phase angle
P = complex(0,phi);
FP = intensityScaling(real(ifft2(exp(P))));
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(FS); title('Reconstructed Image from Spectrum','fontsize',14);
subplot(1,2,2); imshow(FP); title('Reconstructed Image from \phi','fontsize',14);pause(1)
```





The reconstructed images of the girl image are shown above. Using on the intensity information of the spectrum, we see that most of the intensities are centered around the DC component. The reconstructed image from the phase of the girl image displays how vital the phase information is to the image because it carries the shape characteristics of the girl image, which is why you can faintly see the contour of the girl from just the phase info.

(b) Image Reconstruction using Negative Phase

Solution:

```
S1 = S.*exp(1i.*-phi);
Src = revertClass(intensityScaling(real(ifft2(S1))));
figure, imshow(Src), title('Reconstructed Image from Spectrum and -\phi');
```

Reconstructing the image using the negative phase of the girl is equivalent to taking the complex conjugate in the frequency domain. This would result in the image have symmetrically flipped values in the spatial domain, which is why the image appears to be upside down when transforming it back into the spatial domain.

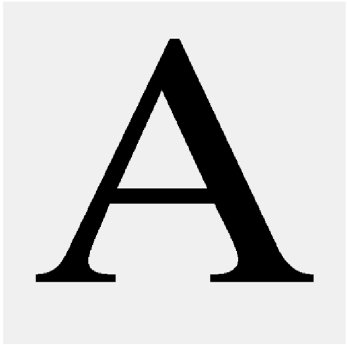
(c) Reconstruction using Phase of another Image.

Solution:

```
% Read both images and display the girl and letter A side-by-side
l = imread('letterA.tif');
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(f); title('Girl Image','fontsize',14);
subplot(1,2,2); imshow(l); title('Letter A Image','fontsize',14);pause(1)
```

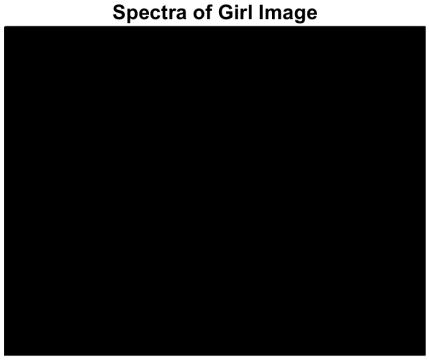
Girl Image

Letter A Image

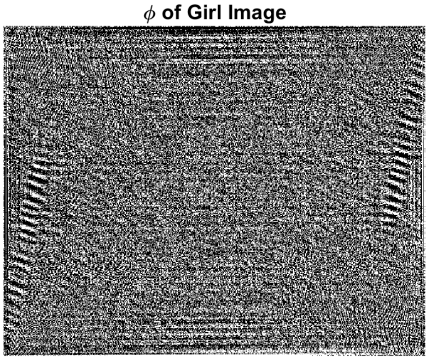



```
% Compute the spectra and phase angles of the letter A image
[l, revertClass] = tofloat(l);
M = max(size(f_girl,1),size(l,1));
N = max(size(f_girl,2),size(l,2));
G = fft2(f,M,N);
L = fft2(l,M,N);
G_mag = abs(G);
G_real = real(G);
G_imag = imag(G);
G_phi = atan2(G_imag,G_real);
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(G_mag,[]); title('Spectra of Girl Image','fontsize',14);
subplot(1,2,2); imshow(G_phi); title('\phi of Girl Image','fontsize',14);pause(1)
```

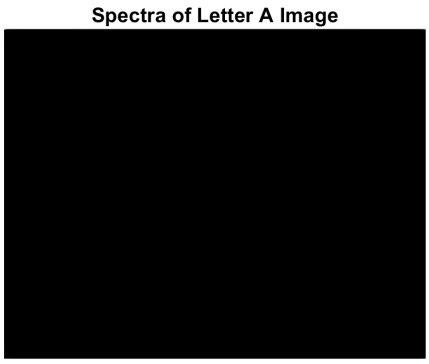
Spectra of Girl Image

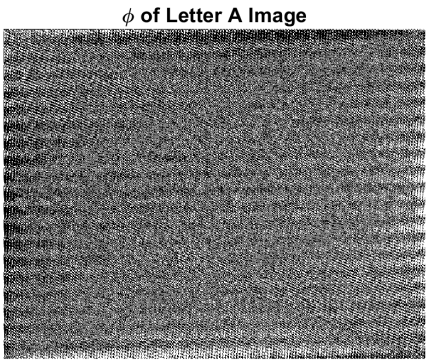


ϕ of Girl Image

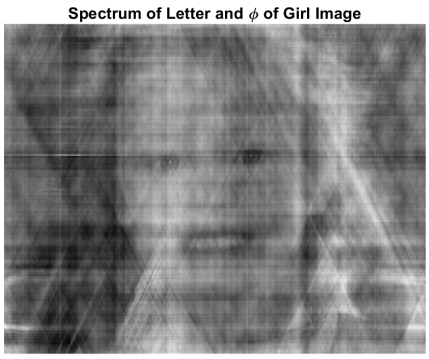


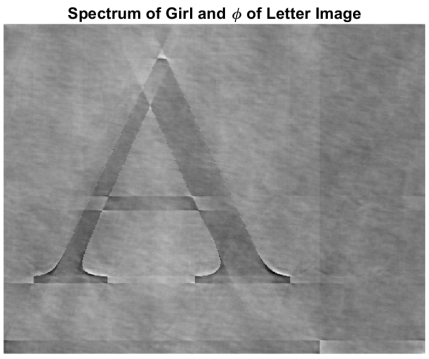
```
L_mag = abs(L); % Magnitude of F(u,v)
L_real = real(L);
L_imag = imag(L);
L_phi = atan2(L_imag,L_real); % Phase of F(u,v)
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(L_mag,[]); title('Spectra of Letter A Image','fontsize',14);
subplot(1,2,2); imshow(L_phi); title('\phi of Letter A Image','fontsize',14);pause(1)
```






```
LSGP = L_mag.*exp(1i.*G_phi);
LSGP_image = revertClass(intensityScaling(real(ifft2(LSGP))));
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1),imshow(LSGP_image),title('Spectrum of Letter and \phi of Girl Image');
GSLP = G_mag.*exp(1i.*L_phi);
GSLP_image = revertClass(intensityScaling(real(ifft2(GSLP))));
subplot(1,2,2), imshow(GSLP_image), title('Spectrum of Girl and \phi of Letter Image');
```





Combing the spectrum of the Letter image and the phase of the Girl image results in a hybrid image that contains the intensity information of the Letter A image and the image shape characteristics of the Girl image. This is why you can still clearly see the image of the girl with faint stripes from the intensity information carried in the Letter image. Likewise, it is the opposite for the hybrid image of the spectrum of the girl and the phase of the letter image. Because these two images were not the same dimensions, a off-centered shifted Letter A is formed from the phase information combined with the intensity information of the girl image. The girl image contained darker intensity values compared to the letter image which this image is mostly gray and contains very little white intensity values. In general, the pixel intensities of these hybrid images are dependent on which image was used for the spectrum information.

Problem-7: DIPUM3E 4.5 (Page 243)

High-Frequency Emphasis (HFE) implementation.

(a) `highfrequencyEmphasis` function

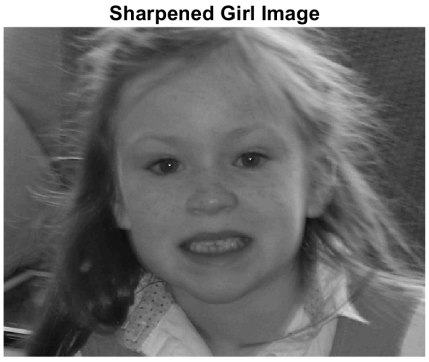
Solution: Insert the code of your function below after the comments.

```
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 filtering 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 also. In
%     the output, G is the processed image and H is the transfer used for
%     high frequency emphasis filtering. Filtering is done using padding,
%     so the filter transfer function is twice the size of F, and 2*DO
%     is used for the cutoff.

% Insert your code below.
% Determine if need to use default values for A,B, and N
if nargin < 3
    n = 2;
    a = 1;
    b = 1;
end
% Create padded dimension of image F
PQ = paddedsize(size(f));
%Use function dftuv to set up meshgrid arrays needed for computing the required distances
[U,V] = dftuv(PQ(1),PQ(2));
%Compute the distances D(U,V)
D = hypot(U,V);
% Cutoff frequency D0 should be twice the input argument of D0
D0 = 2*D0;
% Formula for Butterworth HPF
H = 1./(1 + (D0./D).^(2*n));
% Eq. (4-21) that uses offset A and multiplier B
Hemp = a + b.*H;
gemp = dftfilt(f,Hemp,'symmetric');
g = im2uint8(intensityScaling(gemp));
H = im2uint8(intensityScaling(H));
end
```

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

Solution:

```
f = imread('girl-blurred.tif');
[M,N] = size(f);
D0 = 5;
n = 2.5;
a = 4;
b = 0.2;
[g,H] = highFrequencyEmphasis(f,D0,n,a,b);
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(f); title('Blurred Girl Image','fontsize',14);
subplot(1,2,2); imshow(g); title('Sharpened Girl Image','fontsize',14);pause(1)
```

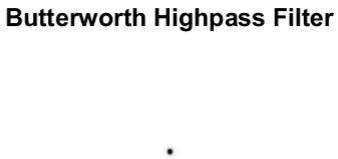


(c) Display of HFE Transfer Function

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

Solution:

```
Hp = H(1:4:end,1:4:end);
Hp = fftshift(Hp);
figure, imshow(Hp), title('Butterworth Highpass Filter')
```



Problem-8 DIPUM3E 4.7 (Page 244)

Sinusoidal Pattern

(a) `impulse2sin` function

Solution: Insert the code of your function below after the comments.

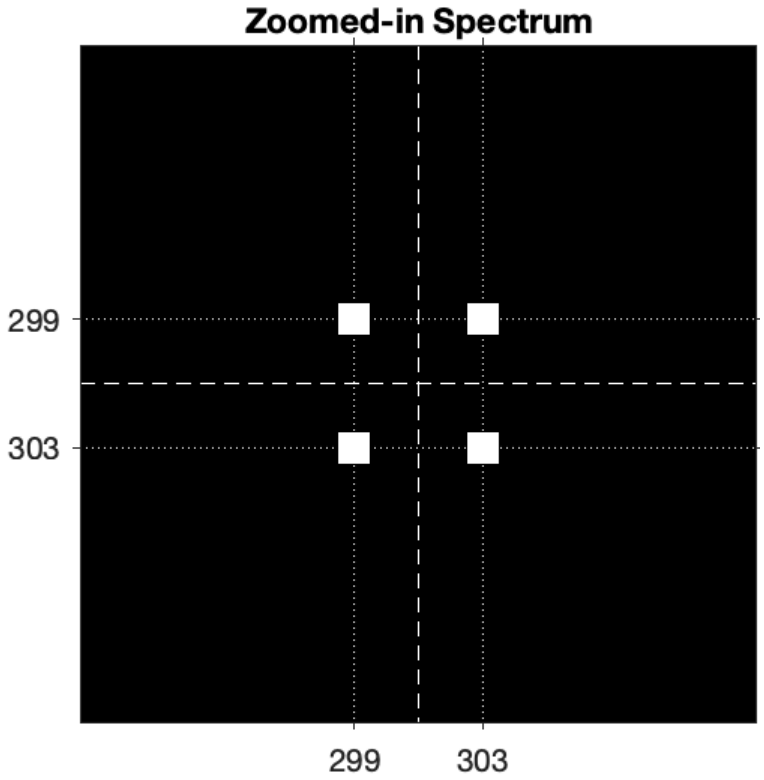
```
function [r,S] = impulses2sin(M,N,C,A)
%IMPULSES2SIN Generates a 2-D spatial  sinusoidal pattern from impulses.

%  [R,S] = IMPULSES2SIN(M,N,C,A), generates a spatial sinusoidal
%      pattern, R, of size M-by-N, and its spectrum, S, in the frequency
```

```
%      domain. The input parameters are as follows:
%
%      C is a K-by-2 matrix with K pairs of frequency domain coordinates
%      (u,v) that define the desired locations of the impulses in the
%      frequency domain. The locations are with respect to the standard
%      origin, at the top left of the rectang1e. Only one pair of (u,v) =
%      (r,c) coordinates is specified for each impu1se. The program
%      computes the location of each conjugate. All impulses must be
%      located in the M-by-N frequency rectangle.
%
%  Because we work with center transforms, the dc term is at
%  coordinates [floor(M/2) + 1, floor(N/2) + 1]. For examp1e, if M = N
%  = 600, the center is at (301,301). If you want to specify an
%  impulse 2 cyc1es away from dc whose corresponding sine wave is
%  oriented in the +45 deg counterclockwise direction with respect to
%  the vertical axis, you cou1d specify C = [299,299], which is in the
%  upper left quadrant with respect to the center. Or, equivalently,
%  you can specify its conjugate at C = [303,303] in the lower 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).

K = size(C,1);
if nargin < 4
    A = ones(1,K);
end
S = zeros(M,N);
%Mag = (1i*M*N)/2;
Center_U = floor(M/2)+1;
Center_V = floor(N/2)+1;
switch K
    case K* (K==1)
       Diff_U1 = Center_U - C(1);
       Diff_V1 = Center_V - C(2);
       if Diff_U1 && Diff_V1 > 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 && Diff_V1 < 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 > 0 && Diff_V1 < 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 < 0 && Diff_V1 > 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 && Diff_V1 == 0
            S(Center_U,Center_V) = A(K);
       end
    case K*(K==2)
       Diff_U1 = Center_U - C(1,1);Diff_U2 = Center_U - C(2,1);
       Diff_V1 = Center_V - C(1,2);Diff_V2 = Center_V - C(2,2);
```

```
        % Shift U1,U2 and V1,V2 of Impulses and Create Complex Conjugates
        if (Diff_U1 && Diff_V1 > 0)
             S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
             S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
        end
        if Diff_U1 && Diff_V1 < 0
             S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
             S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
        end
        if Diff_U1 > 0 && Diff_V1 < 0
             S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
             S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
        end
        if Diff_U1 < 0 && Diff_V1 > 0
             S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
             S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
        end
        if Diff_U1 && Diff_V1 == 0
             S(Center_U,Center_V) = A(K);
        end
        % Shift U2 and V2 of Impulses and Create Complex Conjugates
        if Diff_U2 && Diff_V2 > 0
             S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
             S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 && Diff_V2 < 0
             S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
             S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 > 0 && Diff_V2 < 0
             S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
             S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 < 0 && Diff_V2 > 0
             S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
             S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 && Diff_V2 == 0
             S(Center_U,Center_V) = A(K);
        end
    end
    r = intensityScaling(real(ifft2(fftshift(S))));
    end
```

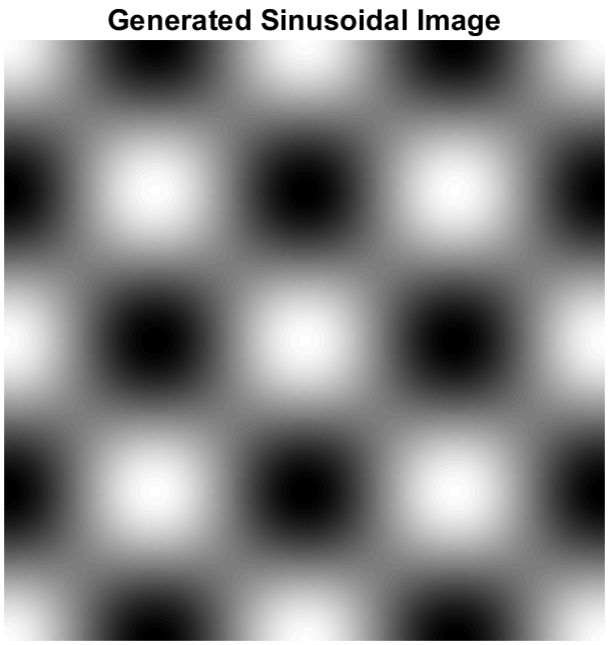
(b) Generation of Sinusoidal Image using `impulse2sin` Function

Note: The image of the spectrum shown in the Student Solutions Manual is incorrect. The zoomed version should look like the following:

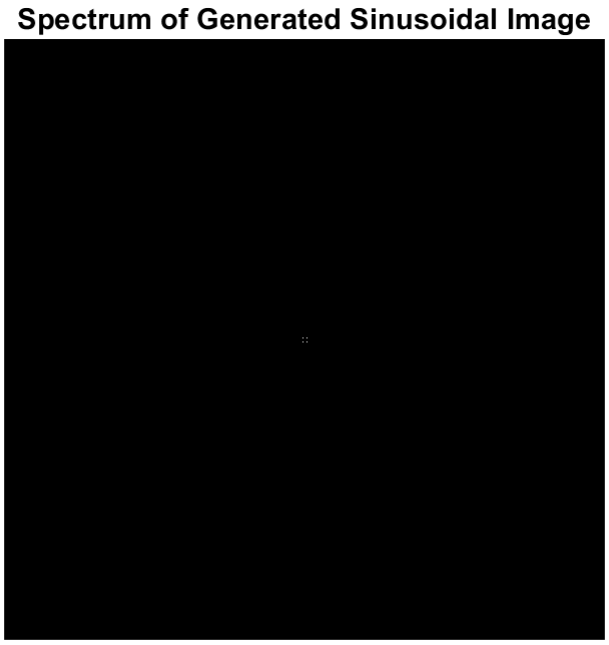


Solution:

```
M = 600;
N = 600;
C = [299,299;303,299];
[r,S] = impulses2sin(M,N,C);
figure, imshow(r),title('Generated Sinusoidal Image')
```



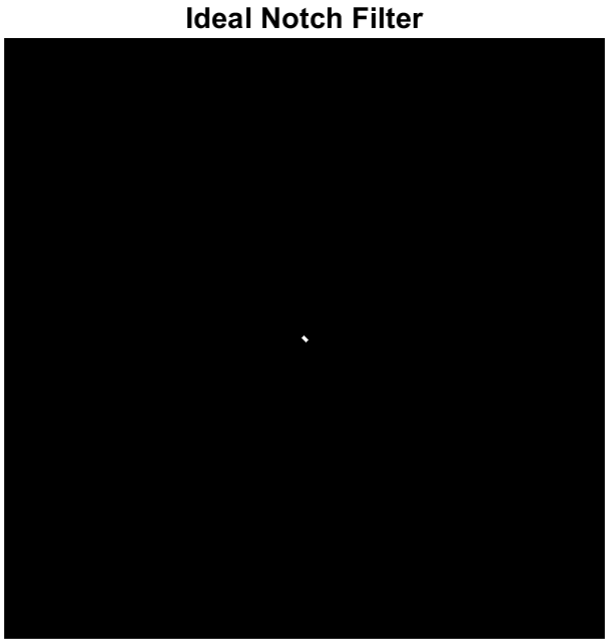

```
figure, imshow(S),title('Spectrum of Generated Sinusoidal Image')
```



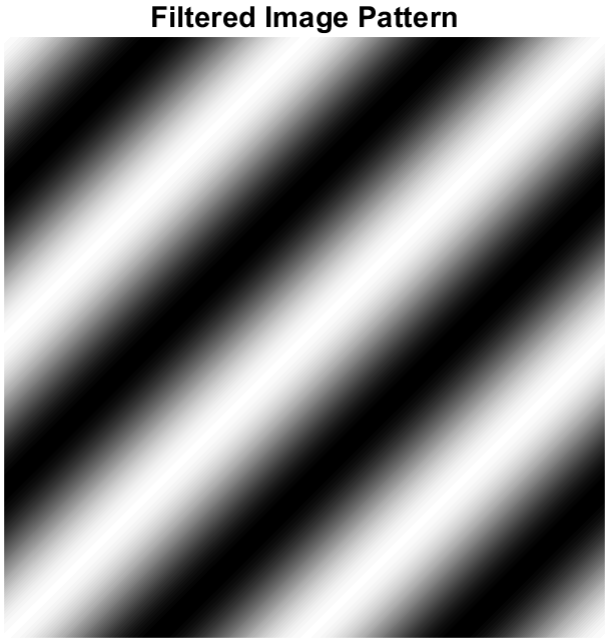
(c) Processing of the above Image

Solution:

```
H = cnotch('ideal','pass',M,N,C,[2,-2]);
g = intensityScaling(dftfilt(r,H));
figure,imshow(fftshift(H)),title('Ideal Notch Filter')
```




```
figure, imshow(g),title('Filtered Image Pattern')
```



```
function H = bpFilterTF4e(type,band,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.

% Insert your code below.

%Protect against uppercase
type = lower(type);
band = lower(band);
%Use function dftuv to set up meshgrid arrays needed for computing the required distances
[U,V] = dftuv(M,N);
%Compute the distances D(U,V)
D = hypot(U,V);
% Determine if need to use default n.
if nargin < 7
    n = 1; %Default BW filter order.
end
%Begin filter computations
switch type
  case 'ideal'
     RI = D <= C0 - (W/2);
     R0 = D >= C0 + (W/2);
     H = tofloat(R0|RI);
  case 'butterworth'
     H = 1./(1 + ((D*W)./(D.^2 - C0^2)).^(2*n));
  case 'gaussian'
     H = 1 - exp(-((D.^2 - C0^2)./(D.*W + eps)).^2);
  otherwise
     error('Unknown filter type.')
end

%Convert to a bandpass transfer function if specified
if isequal(band,'pass')
     H = 1 - H;
end
```

```
end

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 filtering 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 also. In
%     the output, G is the processed image and H is the transfer used for
%     high frequency emphasis filtering. Filtering is done using padding,
%     so the filter transfer function is twice the size of F, and 2*DO
%     is used for the cutoff.

% Insert your code below.
% Determine if need to use default values for A,B, and N
if nargin < 3
    n = 2;
    a = 1;
    b = 1;
end
% Create padded dimension of image F
PQ = paddedsize(size(f));
%Use function dftuv to set up meshgrid arrays needed for computing the required distances
[U,V] = dftuv(PQ(1),PQ(2));
%Compute the distances D(U,V)
D = hypot(U,V);
% Cutoff frequency D0 should be twice the input argument of D0
D0 = 2*D0;
% Formula for Butterworth HPF
H = 1./(1 + (D0./D).^(2*n));
% Eq. (4-21) that uses offset A and multiplier B
Hemp = a + b*H;
gemp = dftfilt(f,Hemp,'symmetric');
g = intensityScaling(gemp);
H = intensityScaling(H);
end

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

%  [R,S] = IMPULSES2SIN(M,N,C,A), generates a spatial sinusoidal
%      pattern, R, of size M—by-N, and its spectrum, S, in the frequency
%      domain. The input parameters are as follows:
%
%      C is a K—by—2 matrix with K pairs of frequency domain coordinates
%      (u,v) that define the desired locations of the impulses in the
%      frequency domain. The locations are with respect to the standard
%      origin, at the top left of the rectang1e. Only one pair of (u,v) =
%      (r,c) coordinates is specified for each impu1se. The program
%      computes the location of each conjugate. All impulses must be
%      located in the M—by-N frequency rectangle.
%
%  Because we work with center transforms, the dc term is at
```

```
%  coordinates [floor(M/2) + 1, floor(N/2) + 1]. For examp1e, if M = N
%  = 600, the center is at (301,301). If you want to specify an
%  impulse 2 cyc1es away from dc whose corresponding sine wave is
%  oriented in the +45 deg counterclockwise direction with respect to
%  the vertical axis, you cou1d specify C = [299,299], which is in the
%  upper left quadrant with respect to the center. Or, equivalently,
%  you can specify its conjugate at C = [303,303] in the lower 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).

K = size(C,1);
if nargin < 4
    A = ones(1,K);
end
S = zeros(M,N);
%Mag = (1i*M*N)/2;
Center_U = floor(M/2)+1;
Center_V = floor(N/2)+1;
switch K
    case K* (K==1)
       Diff_U1 = Center_U - C(1);
       Diff_V1 = Center_V - C(2);
       if Diff_U1 && Diff_V1 > 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 && Diff_V1 < 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 > 0 && Diff_V1 < 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 < 0 && Diff_V1 > 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 && Diff_V1 == 0
            S(Center_U,Center_V) = A(K);
       end
    case K*(K==2)
       Diff_U1 = Center_U - C(1,1);Diff_U2 = Center_U - C(2,1);
       Diff_V1 = Center_V - C(1,2);Diff_V2 = Center_V - C(2,2);
       % Shift U1,U2 and V1,V2 of Impulses and Create Complex Conjugates
       if (Diff_U1 && Diff_V1 > 0)
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
       end
       if Diff_U1 && Diff_V1 < 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
```

```
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
        end
        if Diff_U1 > 0 && Diff_V1 < 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
        end
        if Diff_U1 < 0 && Diff_V1 > 0
            S(Center_U - Diff_U1,Center_V - Diff_V1) = A(K);
            S(Center_U + Diff_U1,Center_V + Diff_V1) = A(K);
        end
        if Diff_U1 && Diff_V1 == 0
            S(Center_U,Center_V) = A(K);
        end
        % Shift U2 and V2 of Impulses and Create Complex Conjugates
        if Diff_U2 && Diff_V2 > 0
            S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
            S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 && Diff_V2 < 0
            S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
            S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 > 0 && Diff_V2 < 0
            S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
            S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 < 0 && Diff_V2 > 0
            S(Center_U - Diff_U2,Center_V - Diff_V2) = A(K);
            S(Center_U + Diff_U2,Center_V + Diff_V2) = A(K);
        end
        if Diff_U2 && Diff_V2 == 0
            S(Center_U,Center_V) = A(K);
        end
end
r = intensityScaling(real(ifft2(fftshift(S))));
end
```