EECE-5626 (IP&PR) : Homework-1

Due on September 24, 2020 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 completing 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 1 and 2 from DIP4E and DSPUM3E.

Problem-1: DIP4E Problem 2.10

Solution:

Assuming the automobile manufacturer provides proper illumination sources within the manufacturing facilities, I would recommend using an RGB sensor, or photodiode, attached to a simple microcontroller like an Arduino to process sensor data from the photodiode. Each color we are trying to determine has a specific energy and wavelength in the visible Electromagnetic spectrum with Blue being 450-495nm, Green being 495-570nm, and Red being the largest at 620-750nm. The RGB sensor uses ratios of the three colors intensities (Red, Green, and Blue) to determine to the color that is being reflected by the object the photodiode is sensing. So, the values for a Red component would have a higher Red intensity from the photodiode, Blue having the most energy of the three would have its highest values in the Blue intensity, and likewise for a Green component and corresponding Green intensity values. Since White light represents light that is balanced in all visible wavelengths, the sensor values of the RGB sensor would be relatively equal for the red, blue, and green intensities. So, colored components would be determined by each individual red, green, and blue intensities and when all three values are close to equal. The price of using an RGB sensor and Arduino would be around $30 at market value. Mechanical motion could also be used like in an assembly line to quickly present the colored component to the sensor and

then sort into a bin or immediately move to another stage where the assembly robots can mount the component to the respective sport cars.

Problem-2: DIP4E Problem 2.13

Solution:

(a) $x + My + MNz = s$

we can then use the mod function on both sides with respect to M to give us

$(x + My + MNz) mod\ M = s \bmod M$

where x must be integer values and the modulus of an integer plus an integer multiple of M will only be equal to the first integer, resulting in

$x = s \bmod M$

(b) $x + My + MNz = s$

to derive the y-coordinate indices we first start by dividing both sides by M to get

$x/M + y + Nz = s/M$

subtract terms to keep y on the left side

$$y = \frac{(s - x)}{M} - \mathrm{Nz}$$

we can then use the mod function on both sides with respect to N to give us

$$y \bmod N = \frac{(s - x)}{M} \bmod N$$

where y must be integer values and the modulus of an integer plus an integer multiple of N will only be equal to the first integer, resulting in

$$y = \frac{(s - x)}{M} \bmod N$$

Problem-3: DIP4E Problem 2.52

Solution:

(a) Suppose that $\{z_k\} = \{2, 5, 3, 3, 1, 0, 1, 0, 4, 4, 5\}$ rearranging and forming the grouping $\{z_k\} = \{(0, 0), (1, 1,), (2), (3, 3,), (4, 4), (5, 5)\}$ we can see the range, $R = \{0, 1, 2, 3, 4, 5\}$

and then $h(z)$ being the number of times the value z occurs in the K observations means $h(0) = 2, h(1) = 2,\ h(2) = 1,\ h(3) = 2,\ h(4) = 2, \text{and}\ h(5) = 2.$

Then using Eq (2-110) we'd get $\bar{z} = \frac{1}{K}\sum_{k=1}^{K} z_k = \frac{1}{11}\sum_{k=1}^{11} z_k = \frac{28}{11}$ and in a similar manner, if we used $h(z)$ we'd get

$$\bar{z} = \frac{1}{K}\sum_{k=1}^{K} z_k = \frac{1}{11}\sum_{j=0}^{5} jh(j) = \frac{1}{11}[(0)(2) + (1)(2) + (2)(1) + (3)(2) + (4)(2) + (5)(2)] = \frac{28}{11}$$

rearranging the latter equation, we see that $\bar{z} = \frac{1}{K}\sum_{k=1}^{K} z_k = \frac{1}{K}\sum_{j\in R} jh(j) = \sum_{j\in R} j\frac{h(j)}{K} = \sum_{j\in R} jp(j)$ which equates Eq (2-92) to Eq (2-110) and results in the same estimate of the mean.

(b) $\sigma^2 = \sum_{z\in R}(z-\bar{z})^2 p(z)$ (Eq. 2-93) and $\sigma^2 = \frac{1}{K}\sum_{k=1}^{K}(z_k - \bar{z})^2$ (Eq. 2-111) can be expanded and using substitution from the previous section

$$\sigma^2 = \sum_{z\in R} z^2 p(z) - \bar{z}^2$$

(c) $\mu_n = E[(z-\bar{z})^n] = \sum_{z\in R}(z-\bar{z})^n p(z)$ (Eq. 2-96)

Problem-4: DIP4E Project 2.10

Solution:

```
% Part A
```

```
function u = centralMoments4e(f,n)
% Computes the mean, variance, and higher-order statistical central
moments
% based on Eq. (2-96) from the DIP4E book. Output u is an n-dimensional
% vector that is computed using p(z), which is the histogram computed
from
% input image f to the nth degree based on input n.

%Initialize output u as an n-dimensional array
u = zeros(1,n);
% Compute Histogram of image g
p = imageHist4e(f);
%Check if intensity values are between [0,1] or [0,255] and compute an
%increment variable that will alter values of z
if max(f(:)) <= 1
    inc = 1/255;
```

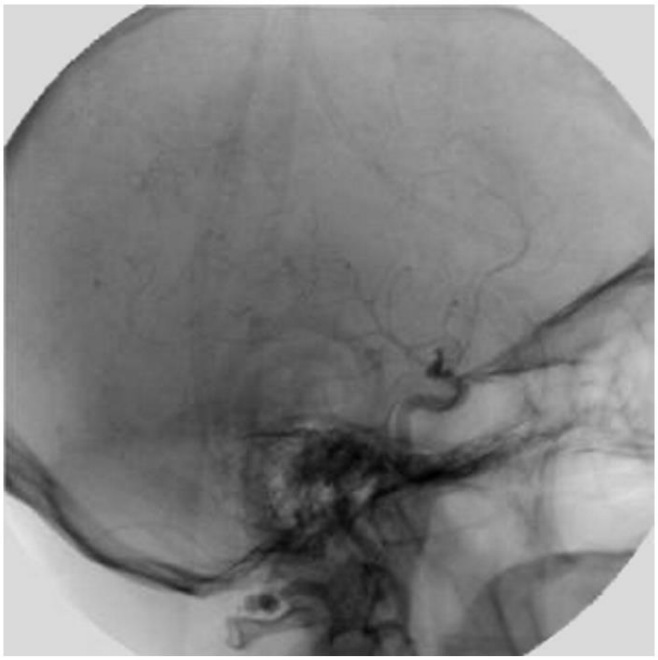
```
    z = 0:inc:1;
else
    inc = 1;
    z = 0:inc:255;
end
%Compute the mean
mean = sum(z.*p);
u(1) = mean;
%Compute remaining moments from Eq. 2-96
for i = 2:n
    u(i) = sum(((z - mean).^i).*p);
end
end
```

```
% Part B
f1 = imread('rose1024.tif');
figure
imshow(f1)
```

```
u1 = centralMoments4e(f1,4);
format shortEng
disp(u1)
```

```
    46.7894e+000     4.3698e+003   503.5215e+003    89.4575e+006
```

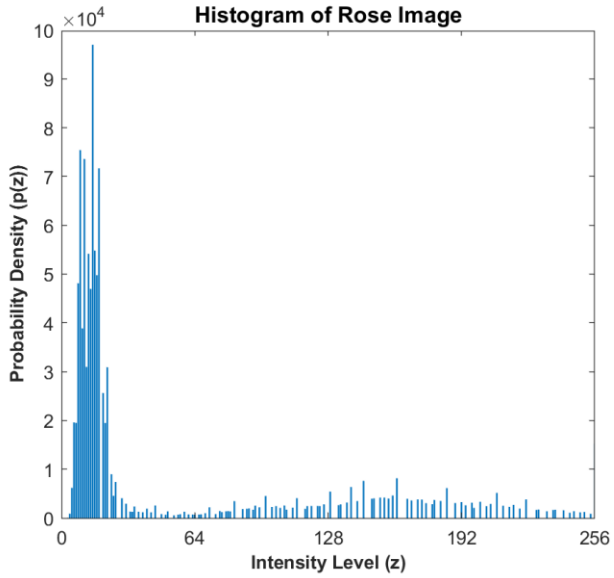
```
% Part C
f2 = imread('angiography-live-image.tif');
figure
imshow(f2)
```



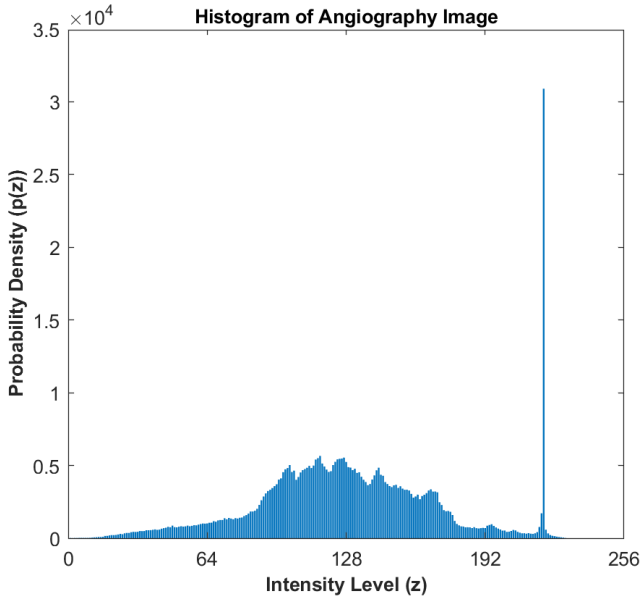
```
u2 = centralMoments4e(f2,4);
display(u2)
```

```
u2 = 1×4
   129.3007e+000     1.8572e+003    14.9847e+003    10.2903e+006
```

```
% Part D
hist1 = imhist(f1);
hist2 = imhist(f2);

bar(hist1,'linewidth',1)
set(gca,'xlim',[0,256],'xtick',(0:64:256));
xlabel('Intensity Level (z)','fontsize',10,'fontweight','bold');
ylabel('Probability Density (p(z))','fontsize',10,'fontweight','bold');
title('Histogram of Rose Image','fontsize',12)
```




```
bar(hist2,'linewidth',1)
set(gca,'xlim',[0,256],'xtick',(0:64:256));
xlabel('Intensity Level (z)','fontsize',10,'fontweight','bold');
ylabel('Probability Density (p(z))','fontsize',10,'fontweight','bold');
title('Histogram of Angiography Image','fontsize',10);
```




```
% Part E
```

The spikes of the histograms represent the number of pixels that represent a certain intensity level with 0 being black and 255 being white. The Rose image had a mean of about 47 indicating that it's average values were darker than most of the total range of intensity and the Angiography image had

a mean of about 129 meaning it contained lighter intensities out of the total range of values. The Rose image also had a higher variance value compared to the second image, which indicates that it had a higher contrast than the Angiography image. The third moment value, or skewness, for the second image was also smaller than the first image and resulted in it's histogram being more symmetrical regardless of the thin spike to the right of the mean. Though, both skewness values were positive meaning which explains why both histogram slightly lean towards the left about the mean. The first image had a larger skewness value and explains the influence of the peaks on the left side of the histogram. Regarding the fourth central moment, the kurtosis, the Rose image had a higher value which indicates why the histogram is flatter than the second image that has more of a bell shape.

Problem-5: DIPUM3E MATLAB Projects 2.3 and 2.6

Solution:

(a) MP 2.3:

```
function [W,S] = iswholeTest(A)
% ISWHOLETEST Elements of an array that are whole numbers (integers)
% [W,S] = ISWHOLETEST(A) returns a logical array, W, of the same size as
A,
% with 1s (TRUE) in the locations corresponding to whole numbers in A,
and
% 0s (FALSE) elsewhere. Scalar S is 1 (TRUE) if ALL elements of A are
whole
% numbers; otherwise S is 0 (FALSE). A must be a real, numeric array. S
is
% used in cases where testing the entire as a unit is required.

L = length(A);
W = logical(zeros(1,L));
for i = 1:L
    if (A(i) == floor(A(i)))
        W(i) = 1;
    elseif (A(i) > floor(A(i)))
        W(i) = 0;
    end
if (sum(W) == length(A))
    S = logical(1);
else
    S = logical(0);
end
end
```

(b) MP 2.6:

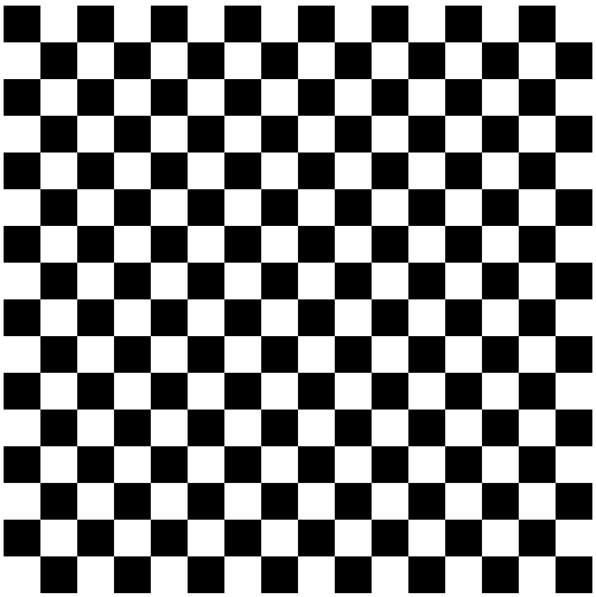
```
% Part A
```

```
function f = testImage(type, param)
%TESTIMAGE Generates gray scale test images.
```

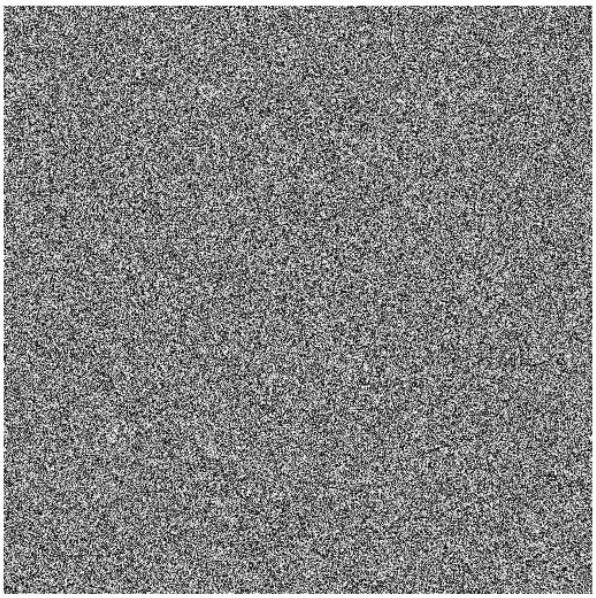
```
% F = TESTIMAGE(TYPE,PARAM) generates a grayscale image of size M-by-N
and
% class double. TYPE is a string, and PARAM is a vector of parameters,
as
% described below:
%
%       TYPE            PARAM               DESCRIPTION
%   'checkerboard'     [M,N,n]          Checkerboard image of size M-by-N
%                                        with each square of size n-by-n
%                                        pixels. Note: Both M N must be
%                                        divisible by 2n; otherwise the
%                                       checkerboard will not make sense.
%   'unoise'           [M,N]            Image of size M-by-N whose pixels
%                                        are uniformly distributed random
%                                        numbers in the range [0,1].
%   'gnoise'           [M,N]           Image of size M-by-N whose pixels
%                              are random numbers drawn from a standard
%                         normal distribution (i.e., a normal (Gaussian)
%                              distribution with mean of 0 and standard
%                                        deviation equal to 1).
%   'vstripe'          [M,N,n]           M-by-N black image with a white,
%                                     centered, vert stripe n pixels wide.
%                                        n cannot exceed N.
%   'hstripe'          [M,N,n]           M-by-N black image with a white,
%                                     centered, horz stripe n pixels wide.
%                                        n cannot exceed M.
t = string(type);
% First parameter 'checkerboard'
if(t == 'checkerboard')
    M = param(1);
    N = param(2);
    n = param(3);
    n2 = 2*n;
    if( iswholeTest(M/(2*n)) ~= 1)
        error('M and N are not divisible by 2n!');
    end
    f  = zeros(M,N);
    f = (checkerboard(n, M/n2, N/n2) > 0.5);

end
% Second parameter 'unoise'
if(t == 'unoise')
    M = param(1);
    N = param(2);
    f = zeros(M,N);
    f = rand(M,N);
end
% Third parameter 'gnoise'
if(t == 'gnoise')
    M = param(1);
    N = param(2);
    mu = 0; sig = 1;
    f = sig.*randn(M,N,'double') + mu;
end
```

```
% Fourth parameter 'vstripe'
if(t == 'vstripe')
    M = param(1);
    N = param(2);
    n = param(3);
    if( n > N)
        error('value of n cannot exceed value of N!')
    end
    f = zeros(M,N);
    f(:,(M/2-n : M/2+n)) = 1;
end
% Fifth parameter 'hstripe'
if(t == 'hstripe')
    M = param(1);
    N = param(2);
    n = param(3);
    if( n > M)
        error('value of n cannot exceed value of M!')
    end
    f = zeros(M,N);
    f((M/2-n : M/2+n),:) = 1;
end
end
```

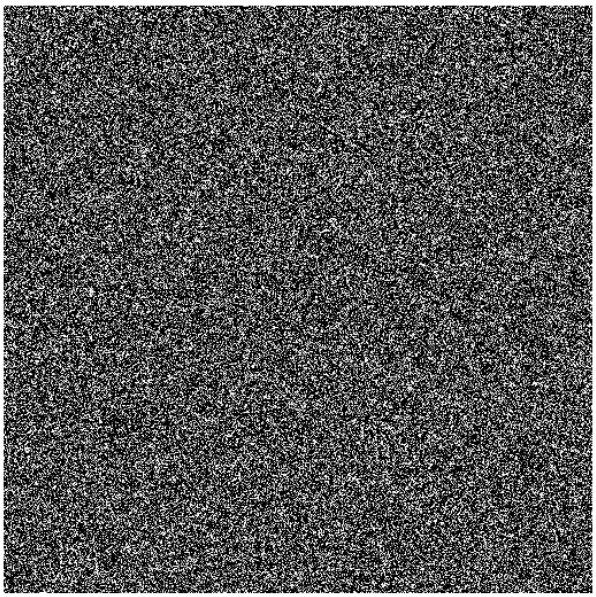
```
% Part B
M = 512;
N = 512;
f_checkerboard = testImage('checkerboard', [M, N, 32]);
figure, imshow(f_checkerboard)
```



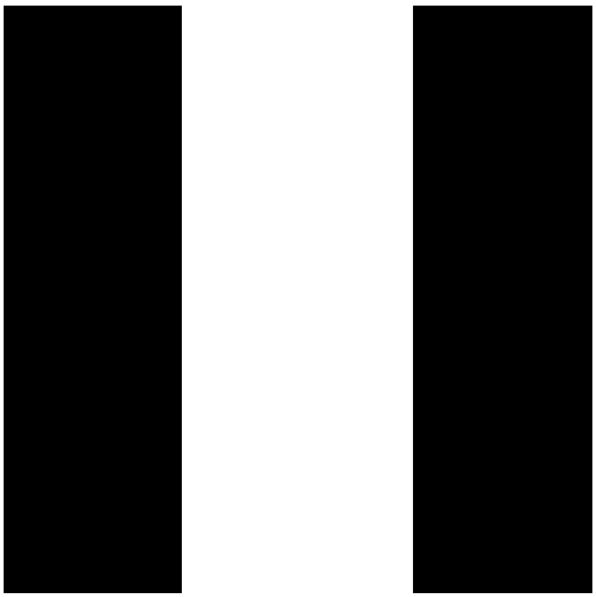
```
f_unoise = testImage('unoise',[M,N]);
figure,imshow(f_unoise)
```



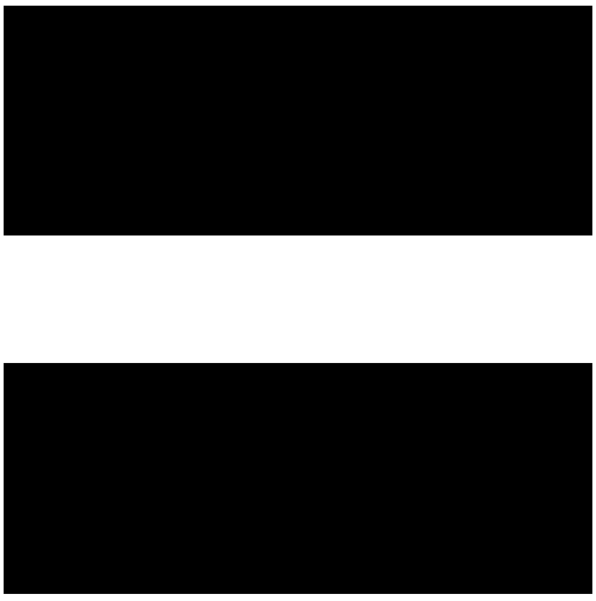
```
f_gnoise = testImage('gnoise',[M,N]);
figure,imshow(f_gnoise)
```



```
f_vert = testImage('vstripe',[M,N, 100]);
figure,imshow(f_vert)
```



```
f_horz = testImage('hstripe',[M,N, 55]);
figure,imshow(f_horz)
```



Problem-6: DIPUM3E MATLAB Project 2.8

Solution:

```
% Part A
```

```
function infoC = myimageinfoC(f)
%MYIMAGEINFOC Numerical information about an image.
%INFO = MYIMAGEINFOC(F) extracts numerical information from image F and
%outputs it using the following cell array forms. In general, F canbe of
% size M-by-N-by-P, with P >= 1 (for example, for RGB images, P = 3).
%
% infoC{1} = number of rows in the image.
% infoC{2} = number of columns.
% infoC{3} = number of image planes.
% infoC{4}(I) = maximum value of each image plane, I = 1:P.
% infoC{5}(I) = minimum value each image plane, I = 1:P.
infoC{1} = size(f,1);
infoC{2} = size(f,2);
infoC{3} = size(f,3);
P = infoC{3};
C_max = {zeros(1:P)};
C_min = {zeros(1:P)};
% Calculate MAX and MIN values of image planes
for i = 1:P
    C_max{i} = max(max(f(:,:,i)));
    C_min{i} = min(min(f(:,:,i)));
end
infoC{4} = C_max;
infoC{5} = C_min;
end
```

```
% Part B
f = imread('ballerina.tif');
figure, imshow(f)
```

```
infoC = myimageinfoC(f)
```

infoC = 1×5 cell

	1	2	3	4	5
1	644	656	3	1×3 cell	1×3 cell

```
num_rows = infoC{1}
```

num_rows = 644

```
num_cols = infoC{2}
```

num_cols = 656

```
num_image_planes = infoC{3}
```

num_image_planes = 3

```
max_img_planes = infoC{4}
```

max_img_planes = 1×3 cell

	1	2	3
1	255	255	255

```
min_img_planes = infoC{5}
```

min_img_planes = 1×3 cell

	1	2	3
1	0	0	1

```
format short
```

Problem-7

Let $f(m,n)$ be an input image, $h(m,n)$ be an impulse response, and $g(m,n)$ be the corresponding output image obtained via the 2-D convolution. Prove the following volume conservation property:

$$\sum_{m,n=-\infty}^{\infty}\sum g(m,n) = \left[\sum_{m,n=-\infty}^{\infty}\sum h(m,n)\right]\left[\sum_{m,n=-\infty}^{\infty}\sum f(m,n)\right]$$

Proof:

$$\begin{aligned}
\sum_{m,n=-\infty}^{\infty}\sum g(m,n) &= \sum_{m,n=-\infty}^{\infty}\sum [h(m,n) * f(m,n)] \\
&= \sum_{m,n=-\infty}^{\infty}\sum \left[\sum_{k,l=-\infty}^{\infty}\sum h(k,l) f(m-k,n-l)\right] \\
&= \sum_{k,l=-\infty}^{\infty}\sum \left[\sum_{m,n=-\infty}^{\infty}\sum h(k,l) f(m-k,n-l)\right] \\
&= \sum_{k,l=-\infty}^{\infty}\sum h(k,l) \left[\sum_{m,n=-\infty}^{\infty}\sum f(m-k,n-l)\right] \\
&= \left[\sum_{k,l=-\infty}^{\infty}\sum h(k,l)\right]\left[\sum_{m,n=-\infty}^{\infty}\sum f(m,n)\right] \\
&= \left[\sum_{m,n=-\infty}^{\infty}\sum h(m,n)\right]\left[\sum_{m,n=-\infty}^{\infty}\sum f(m,n)\right]
\end{aligned}$$

Problem-8

Consider the following two-dimensional signal $f(m,n)$, where the bracketed element denotes the $(0,0)$ location

$$f(m,n) = \begin{matrix} 8 & 5 & 3 \\ 6 & 7 & 4 \\ 4 & [7] & 6 \\ 3 & 5 & 8 \end{matrix} \begin{matrix} \\ \\ \rightarrow n \\ \\ \downarrow \\ m \end{matrix}$$

(a) Determine the convolution of $f(m,n)$ with each of the following signals

i.
$$\begin{matrix} 0 & 1 & 0 \\ 1 & [-4] & 1 \\ 0 & 1 & 0 \end{matrix}$$

Solution :

```
f1 = [8, 5, 3; 6, 7, 4; 4, 7, 6; 3, 5, 8];
h1 = [0, 1, 0; 1, -4, 1; 0, 1, 0];
g1 = conv2(f1,h1)
```

```
g1 = 6×5
     0     8     5     3     0
     8   -21    -2    -3     3
     6    -5    -6     0     4
     4     0    -6    -5     6
     3    -3    -2   -21     8
     0     3     5     8     0
```

ii.
$$\begin{matrix} 1 & 0 & 0 \\ 0 & [2] & 0 \\ 0 & 0 & 1 \end{matrix}$$

Solution :

```
f2 = [8, 5, 3; 6, 7, 4; 4, 7, 6; 3, 5, 8];
h2 = [1, 0, 0; 0, 2, 0; 0, 0, 1];
g2 = conv2(f2,h2)
```

```
g2 = 6×5
     8     5     3     0     0
     6    23    14     6     0
     4    19    28    13     3
     3    13    28    19     4
     0     6    14    23     6
     0     0     3     5     8
```

iii.
$$\begin{matrix} 0 & 2 \\ [-2] & 0 \\ 0 & 2 \end{matrix}$$

Solution :

```
f3 = [8, 5, 3; 6, 7, 4; 4, 7, 6; 3, 5, 8];
h3 = [0, 2; -2, 0; 0, 2];
g3 = conv2(f3,h3)
```

```
g3 = 6×4
     0    16    10     6
   -16     2     8     8
   -12    10    16    18
    -8     4    12    24
    -6    -2    -2    12
     0     6    10    16
```

(b) Verify your answers in each case via the ***volume conservation*** property proved in **Problem 7** above.

Solution :

```
% For each convolution operated above, the resulting g(m,n) should equal to
% the sum of the product of h(m,n) and f(m,n)

% First Convolution
sum_f1 = sum(sum(f1)); sum_h1 = sum(sum(h1)); sum_g1 = sum(sum(g1));
if(sum_g1 == sum_h1 * sum_f1)
    disp('volume conservation property verified')
end
```

```
volume conservation property verified
```

```
% Second Convolution
sum_h2 = sum(sum(h2)); sum_g2 = sum(sum(g2));
if(sum_g2 == sum_h2 * sum_f1)
    disp('volume conservation property verified')
end
```

```
volume conservation property verified
```

```
% Third Convolution
sum_h3 = sum(sum(h3)); sum_g3 = sum(sum(g3));
if(sum_g3 == sum_h3 * sum_f1)
    disp('volume conservation property verified')
end
```

```
volume conservation property verified
```

Functions

P4 - DIP4E 2.10 function

```
function u = centralMoments4e(f,n)
```

```
% Computes the mean, variance, and higher-order statistical central moments
% based on Eq. (2-96) from the DIP4E book. Output u is an n-dimensional
% vector that is computed using p(z), which is the histogram computed from
% input image f to the nth degree based on input n.

%Initialize output u as an n-dimensional array
u = zeros(1,n);
% Compute Histogram of image g
p = imageHist4e(f);
%Check if intensity values are between [0,1] or [0,255] and compute an
%increment variable that will alter values of z
if max(f(:)) <= 1
    inc = 1/255;
    z = 0:inc:1;
else
    inc = 1;
    z = 0:inc:255;
end
%Compute the mean
mean = sum(z.*p);
u(1) = mean;
%Compute remaining moments from Eq. 2-96
for i = 2:n
    u(i) = sum(((z - mean).^i).*p);
end
end
```

P5 - DIPUM3E Matlab Project 2.3 function

```
function [W,S] = iswholeTest(A)
% ISWHOLETEST Elements of an array that are whole numbers (integers)
% [W,S] = ISWHOLETEST(A) returns a logical array, W, of the same size as A,
% with 1s (TRUE) in the locations corresponding to whole numbers in A, and
% 0s (FALSE) elsewhere. Scalar S is 1 (TRUE) if ALL elements of A are whole
% numbers; otherwise S is 0 (FALSE). A must be a real, numeric array. S is
% used in cases where testing the entire as a unit is required.

L = length(A);
W = logical(zeros(1,L));
for i = 1:L
    if (A(i) == floor(A(i)))
        W(i) = 1;
    elseif (A(i) > floor(A(i)))
        W(i) = 0;
    end
if (sum(W) == length(A))
```

```
    S = true;
else
    S = logical(0);
end
end
end
```

P5 - DIPUM3E Matlab Project 2.6 function

```
function f = testImage(type, param)
%TESTIMAGE Generates gray scale test images.
% F = TESTIMAGE(TYPE,PARAM) generates a grayscale image of size M-by-N and
% class double. TYPE is a string, and PARAM is a vector of parameters, as
% described below:
%
%       TYPE            PARAM              DESCRIPTION
%   'checkerboard'     [M,N,n]          Checkerboard image of size M-by-N
%                                       with each square of size n-by-n
%                                       pixels. Note: Both M N must be
%                                       divisible by 2n; otherwise the
%                                       checkerboard will not make sense.
%   'unoise'           [M,N]            Image of size M-by-N whose pixels
%                                       are uniformly distributed random
%                                       numbers in the range [0,1].
%   'gnoise'           [M,N]            Image of size M-by-N whose pixels
%                                       are random numbers drawn from a
standard
%                                       normal distribution (i.e., a normal
(Gaussian)
%                                       distribution with mean of 0 and
standard
%                                       deviation equal to 1).
%   'vstripe'          [M,N,n]          M-by-N black image with a white,
%                                       centered, vert stripe n pixels wide.
%                                       n cannot exceed N.
%   'hstripe'          [M,N,n]          M-by-N black image with a white,
%                                       centered, horz stripe n pixels wide.
%                                       n cannot exceed M.
t = string(type);
% First parameter 'checkerboard'
if(t == 'checkerboard')
    M = param(1);
    N = param(2);
    n = param(3);
    n2 = 2*n;
```

```
    if( iswholeTest(M/(2*n)) ~= 1)
        error('M and N are not divisible by 2n!');
    end
    f  = zeros(M,N);
    f = (checkerboard(n, M/n2, N/n2) > 0.5);

end
% Second parameter 'unoise'
if(t == 'unoise')
    M = param(1);
    N = param(2);
    f = zeros(M,N);
    f = rand(M,N);
end
% Third parameter 'gnoise'
if(t == 'gnoise')
    M = param(1);
    N = param(2);
    mu = 0; sig = 1;
    f = sig.*randn(M,N,'double') + mu;
end
% Fourth parameter 'vstripe'
if(t == 'vstripe')
    M = param(1);
    N = param(2);
    n = param(3);
    if( n > N)
        error('value of n cannot exceed value of N!')
    end
    f = zeros(M,N);
    f(:,(M/2-n : M/2+n)) = 1;
end
% Fifth parameter 'hstripe'
if(t == 'hstripe')
    M = param(1);
    N = param(2);
    n = param(3);
    if( n > M)
        error('value of n cannot exceed value of M!')
    end
    f = zeros(M,N);
    f((M/2-n : M/2+n),:) = 1;
end
end
```

P6: DIPUM3E Matlab Project 2.8 function

```
function infoC = myimageinfoC(f)
%MYIMAGEINFOC Numerical information about an image.
% INFO = MYIMAGEINFOC(F) extracts numerical information from image F and
% outputs it using the following cell array forms. In general, F can be of
% size M-by-N-by-P, with P >= 1 (for example, for RGB images, P = 3).
%
% infoC{1} = number of rows in the image.
% infoC{2} = number of columns.
% infoC{3} = number of image planes.
% infoC{4}(I) = maximum value of each image plane, I = 1:P.
% infoC{5}(I) = minimum value each image plane, I = 1:P.
infoC{1} = size(f,1);
infoC{2} = size(f,2);
infoC{3} = size(f,3);
P = infoC{3};
C_max = {zeros(1:P)};
C_min = {zeros(1:P)};
% Calculate MAX and MIN values of image planes
for i = 1:P
    C_max{i} = max(max(f(:,:,i)));
    C_min{i} = min(min(f(:,:,i)));
end
infoC{4} = C_max;
infoC{5} = C_min;
end
```