```
%%Exercise 3.1
% USER DEFINED VARIABLES

w = 15; % Width
x = 1:160; % Horiztonal Axis
y = 1:80; % Vertical Axis

% == > Creates a Matrix with the parameters from the above <==
z = round (127* exp ( -1/ w .^2*(( y .'-40) .^2+( x -80) .^2) ) );

% == > changes the colormap matrix to the gray scale<==
colormap ( gray ) ;

% == > Image system 1 stretches the image, image system 2 moves the
 image
% to the lower left, image system 3 makes the image split to the 4
 corners
%of the figure<==
[ xs , ys , zs ] = image_system1 (z ,2 ,2) ;
za               = image_system2 ( zs , -10 ,35) ;
zb               = image_system3 ( za , -30 ,35) ;

% PLOT RESULT WITH SUBPLOT

figure (1) ;
subplot (2 ,2 ,1) ; % == > puts plot in 1st box of 4x4 figure matrix
 <==
imagesc (x , y , z ) ; % == > plots the image matrix z in a x by y
 plot <==
axis image ; % == > plot box fits around data <==
title ( 'Original ')
subplot (2 ,2 ,2) ; % == > puts plot in 2nd box of 4x4 figure
 matrix<==
imagesc ( xs , ys , zs ) ; % == > plots image Zs in xs and ys axis <==
axis image ; % == > plot box fits around image<==
title ( ' After System 1 ')
subplot (2 ,2 ,3) ; % == > Plots data in 3rd box of 4x4 figure matrix
 <==
imagesc ( xs , ys , za ) ; % == > plots za on xs and ys axis<==
axis image ; % == > fits plot box to island <==
title ( ' After System 2 ')
subplot (2 ,2 ,4) ; % == > Plots in the 4th box of the 4x4 figure
 matrix <==
imagesc ( xs , ys , zb ) ; % == > Plots zb in the xs and ys  <==
axis image ; % == > fits plot box to plot data <==
title ( ' After System 3 ')

%%Exercise 3.2
%b
colormap('gray')
lighthouse = load('lighthouse.mat');
x = 0:1:size(lighthouse.lighthouse,2);
```

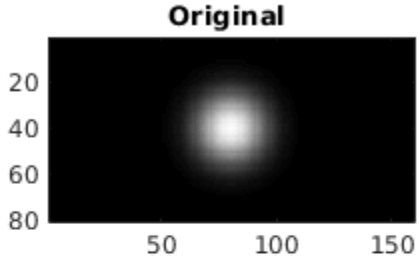
```
y = 0:1:size(lighthouse.lighthouse,1);
[xs , ys, lighthouse_sampled] = image_sample(lighthouse.lighthouse,
 2);

figure (2) ;
subplot (2 ,2 ,1) ;
imagesc (x , y , lighthouse.lighthouse ) ;
axis image ;
title ( 'Original ')

subplot (2 ,2 ,2) ;
imagesc (xs , ys , lighthouse_sampled ) ;
axis image ;
title ( 'sampled ')

%the sampling makes the image blurrier, and the image loses some of
 its
%details. This relates to how signals lose information when
 undersampled
lighthouse_aax6 = lighthouse.lighthouse;

%d

for k = 1:6
    lighthouse_aax6 = image_antialias(lighthouse_aax6);
end

subplot (2 ,2 ,3) ;
imagesc (x , y , lighthouse_aax6) ;
axis image ;
title ( 'Anti-Aliased ')

[xs, ys ,lighthouse_aax6_sampled] = image_sample(lighthouse_aax6,2);

subplot (2 ,2 ,4) ;
imagesc (xs , ys , lighthouse_aax6_sampled) ;
axis image ;
title ( 'Anti-Aliased Sampled')

%f

[xz,yz,lighthouse_zeros] =
 image_insertzeros(lighthouse_aax6_sampled,2);

for k = 1:6
    lighthouse_interpolated = image_antialias(lighthouse_zeros);
end

% The dimensions of the interpolated image is the same as the original
figure(3);
colormap('gray')
subplot (2 ,2 ,1) ;
imagesc (x , y , lighthouse.lighthouse ) ;
axis image ;
```

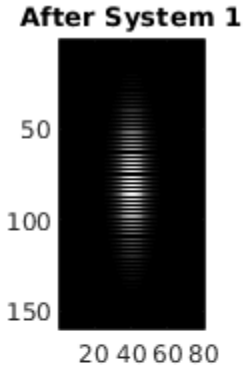
```
title ( 'Original ')

subplot (2 ,2 ,2) ;
imagesc (xz , yz , lighthouse_zeros ) ;
axis image ;
title ( 'zeros ')

subplot (2 ,2 ,3) ;
imagesc (xz , yz , lighthouse_interpolated ) ;
axis image ;
title ( 'interpolated')

% The interpolation filter smoothes the image out and creates the
 illusion
% of recovered information. This is useful in smoothing out signals
 for
% user reception

%%FUNCTIONS USED
type 'image_system2'
type 'image_system1'
type 'image_system3'
type 'image_insertzeros'
type 'image_sample'
type 'image_antialias'



function [ za ] = image_system2 (z , Sx , Sy )
% IMAGE_SYSTEM2 === > Moves the image to the lower left<===
% ==== > Creates a zeros matrix of the image size <====
za = zeros ( size (z ,1) , size (z ,2) ) ;
for nn = 1: size (z ,1)
for mm = 1: size (z ,2)
% ==== > If the image is not already where it is to be shifted <====
if nn > Sy && nn - Sy < size(z ,1) && mm > Sx && mm - Sx < size(z ,2)
    % ==== > Shift the image <====
    za ( nn , mm ) = 1/2* z ( nn - Sy , mm - Sx ) ;
end
end
end
end

function [ xs , ys , zs ] = image_system1 (z , Dx , Uy )
% IMAGE_SYSTEM1 === > This function changes the image by stretching it
%by a factor of Uy and shrinking it by a factor of Dx<===
% == > Creates a Zeros matrix of the size  <==
zs = zeros ( ceil ( Uy * size (z ,1) ) , ceil ( size (z ,2) / Dx ) ) ;
% == > Creates modified X and Y axis <==
ys = 1: ceil ( Uy * size (z ,1) ) ;
xs = 1: ceil ( size (z ,2) / Dx ) ;
% == > instantiates the new zs matrix with the modified dimensions<==
zs (1: Uy : end ,1: end ) = z (1: end ,1: Dx : end ) ;
end
```

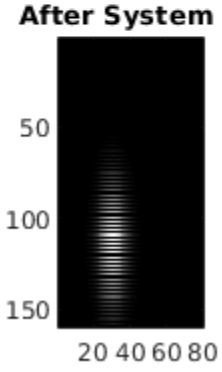
```
function [ zb ] = image_system3 ( za , Sx , Sy )
% IMAGE_SYSTEM3 === > This function overlaps the image across axis
%boundries<===
% ==== > Create new axis<====
x = 0:1: size ( za ,2) -1;
y = 0:1: size ( za ,1) -1;
% ==== > Rescale with mod operator <====
xs = mod (x - Sx , size ( za ,2) ) ;
ys = mod (y - Sy , size ( za ,1) ) ;
% ==== > Instantiate the image matrix with the new info <====
zb = za ( ys +1 , xs +1) ;
end

function [xz, yz, zz] = image_insertzeros(zaas, U)
insertXSize = ((size(zaas,2) - 1) * (U-1)) + size(zaas,2);
insertYSize = ((size(zaas,1) - 1) * (U-1)) + size(zaas,1);
zz = zeros(insertYSize,insertXSize);
i = 1;
k = 1;
zaasi = 1;
zaask = 1;
while(i <= (insertYSize))
   while (k <= (insertXSize))
        if ((mod(i,2) == 0) || (mod(k,2) == 0))
              %  zz(i,k) = 0;
               % k = k+1;
               % zaask = zaask+1;
               zz(i,k) = zaas(zaasi,zaask);
               k = k+1;
        else
         %   zz(i,k) = zaas(zaasi,zaask);
           % k = k+1;
               zz(i,k) = 0;
               k = k+1;
               zaask = zaask+1;

        end

   end
      i = i+1;
      k = 1;
      if(mod(i,2) == 0)
        zaasi = zaasi + 1;
      end
      zaask = 1;
end
xz = 0:1:insertXSize;
yz = 0:1:insertYSize;
end
```

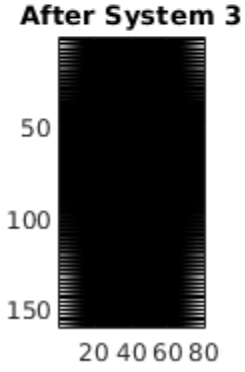
```
function [xs , ys , zs ] = image_sample (z , D)
x = zeros(ceil(size(z,2)/D),1); % size of aliased x axis
y = zeros(ceil(size(z,1)/D),1);%size of aliased y axis
zs = zeros(size(y,1),size(x,1)); %size of alaised image
xs = 0:1:size(x); %x axis
ys = 0:1:size(y); %y axis
for i = 1:size(x) %aliasing
    for k = 1:size(y)
        zs(k,i) = z(k*D,i*D);
    end
end
```

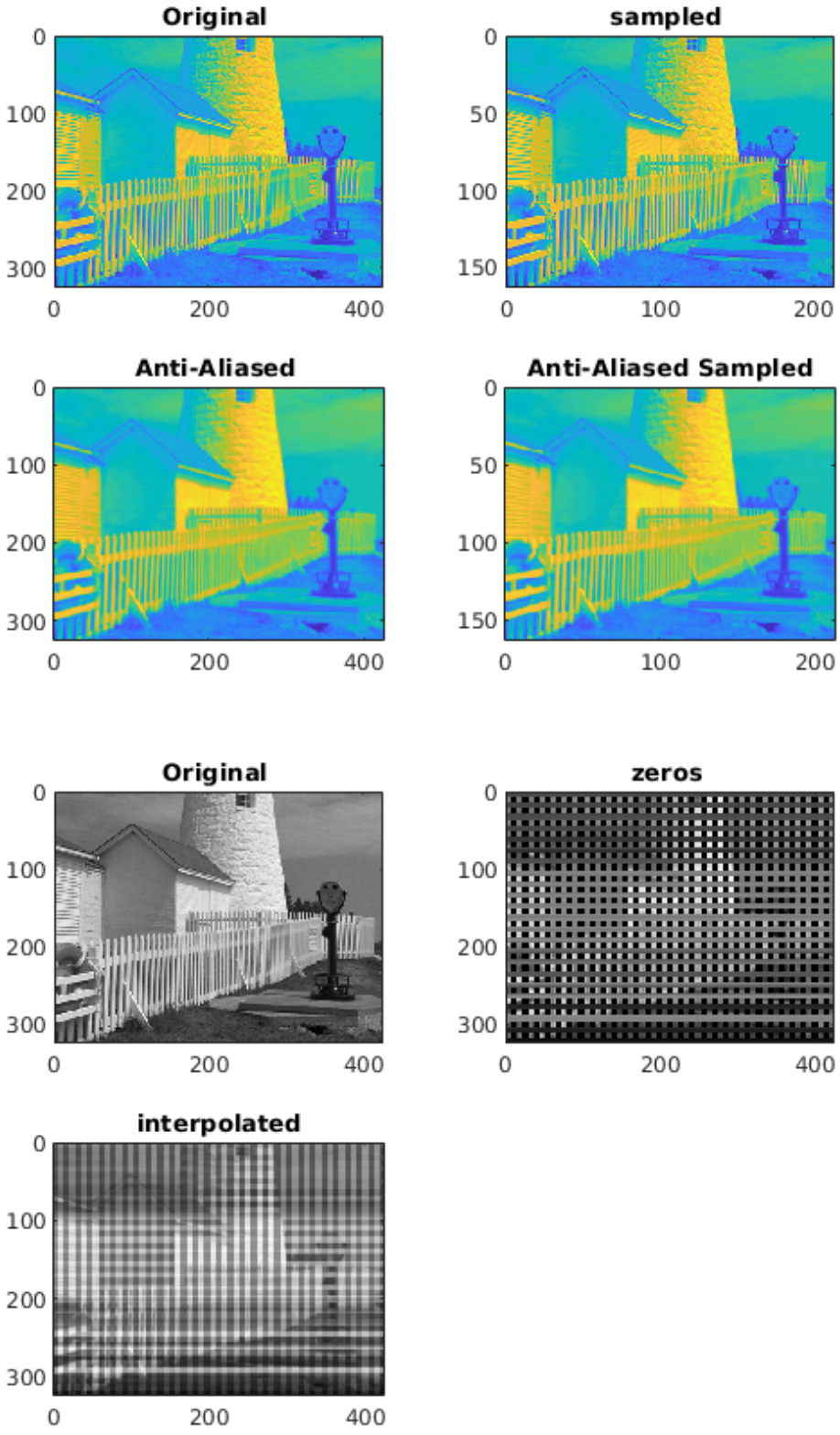
```
function zaa = image_antialias(z)
zaa = zeros(size(z,1),size(z,2));
for i = 2:size(z,1)-1
    for k = 2:size(z,2)-1
        zaa(i,k) = (1/2) * z(i,k) + (1/8) * (z(i-1,k) +z(i+1,k) + ...
            z(i,k-1) + z(i, k+1));
    end
end
```


Original
20
40
60
80
50
100
150


After System 1
50
100
150
20 40 60 80


After System 2
50
100
150
20 40 60 80


After System 3
50
100
150
20 40 60 80


Original
sampled
Anti-Aliased
Anti-Aliased Sampled
Original
zeros
interpolated

Published with MATLAB® R2020a