EE386 Lab 3 - RGB, Grayscale, Sampling, and Reconstruction

Nov. 5, 2018

Lab directed by Chris Hirunthanakorn

Lab conducted by Paco Ellaga

CSULB SID: 009602331

- Double - A type of image where its range values are between [0, 1]. It also can be switched into uint8 and vice versa.
- Uint8 - A type of image where the range of its capacities stays within 0 to 2^8-1 (255) pixels. It is also short for unsigned 8 bit integer.
- Indexed - It is composed of an array(X) and a colormap(mp) that pick the red, green, and blue components to create a color.
- Binary - With each pixel being dependent on either being a value of 1 or 0, the logical array it is stored in will create an image while using the variable name, "BW".
- Intensity - A range of color scaled arrays where double provides [0,1], 8-bit provides [0, 255], and 16-bit provides [0 65535].

PART 1

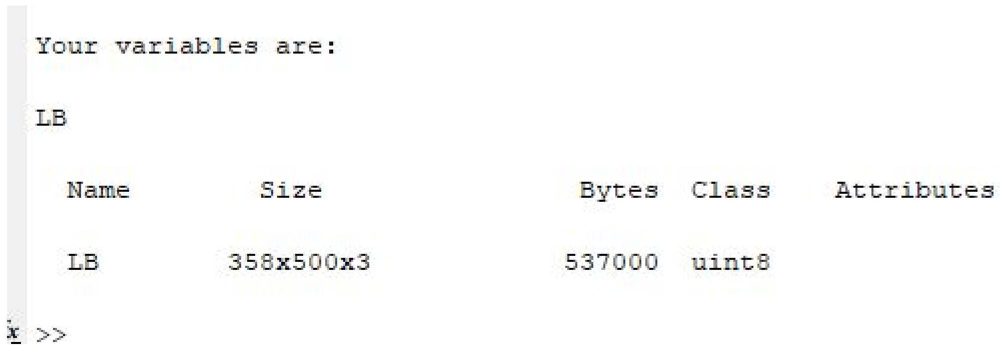
%% Part 1: Manipulating Long Beach Harbor picture (File renamed to "LB")

Question A, B:

The size of the picture is 358x500x3. The type is a .jpeg of a uint8.

```
who                         % lists current variable (image)
whos
```

As of result and at time of the *who* command, the following show up in result of the Long Beach Harbor:



```
Your variables are:

LB

  Name        Size                Bytes  Class    Attributes

  LB        358x500x3            537000  uint8

>>
```

Figure 1: *Variables when loading LB at after uiopen().*

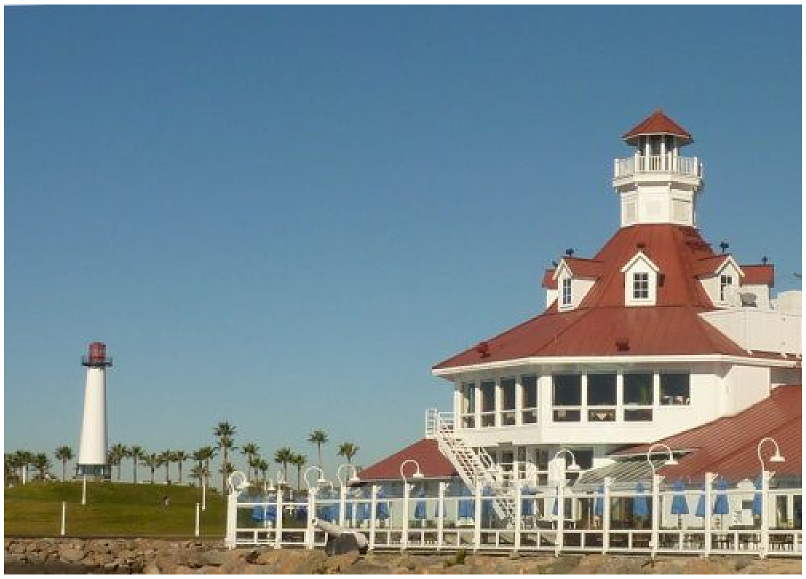


Figure 2: *Original picture of the Long Beach Harbor (renamed to LB in project).*

<u>Question C, D</u>

As result of the grayscale, the size of the new image is 358x500. When changing to gray the last dimension of x3 is removed because it now only in two dimensions. The x3 is the dimension representing depth, while the first value (358) is width and the second (500) is length.

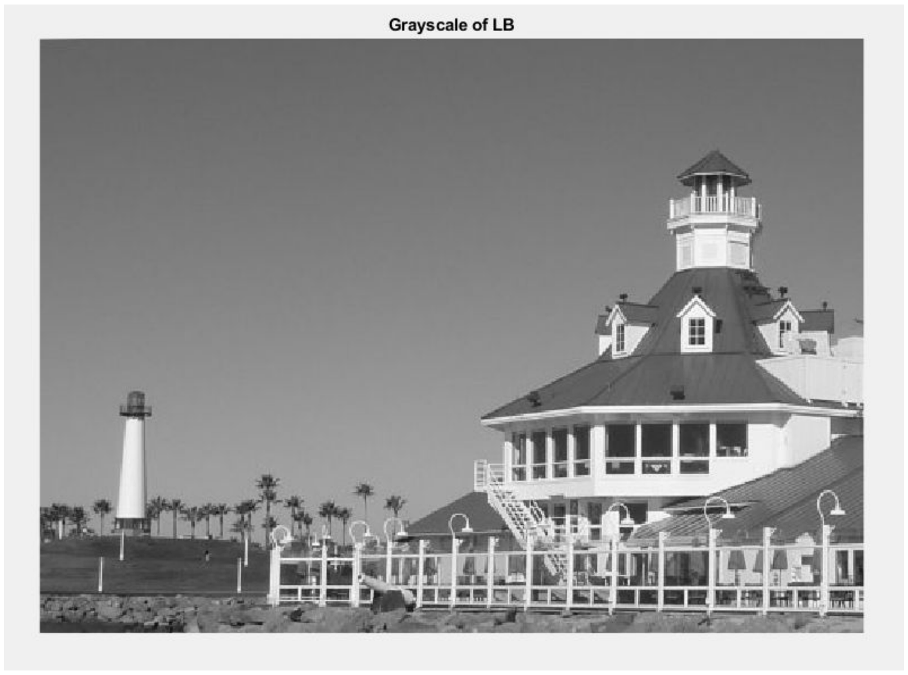


Figure 3: *GrayScale of LB.*

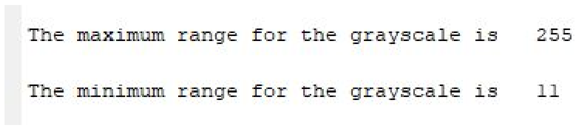
Question K

Next, the maximum and minimum range was to be identified. This was completed and identified through the following commands set below:

```
LBbw_Max = max(max(LBbw));      % finds max value
fprintf('The maximum range for the grayscale is'),disp(LBbw_Max)

LBbw_Min = min(min(LBbw));      % finds min value
fprintf('The minimum range for the grayscale is'),disp(LBbw_Min)
```

As return of the the two *fprintf()* commands, this is the result:



```
The maximum range for the grayscale is   255

The minimum range for the grayscale is   11
```

Figure 4: *The max and min range values in grayscale 8-bit format*

Question E, F

The values of all 3 (red, green, blue) are the same with 357x500. But are missing the "x3" at the end, like the grayscale. This is because rgb relates to [1 1 1] where they are [red green blue]. When removed, the channel reflect on the removal. The Red channel has the lightest portion in where the red used to be. The green has the removal of the plains and the blue has removal of the tint in the sky and the building.

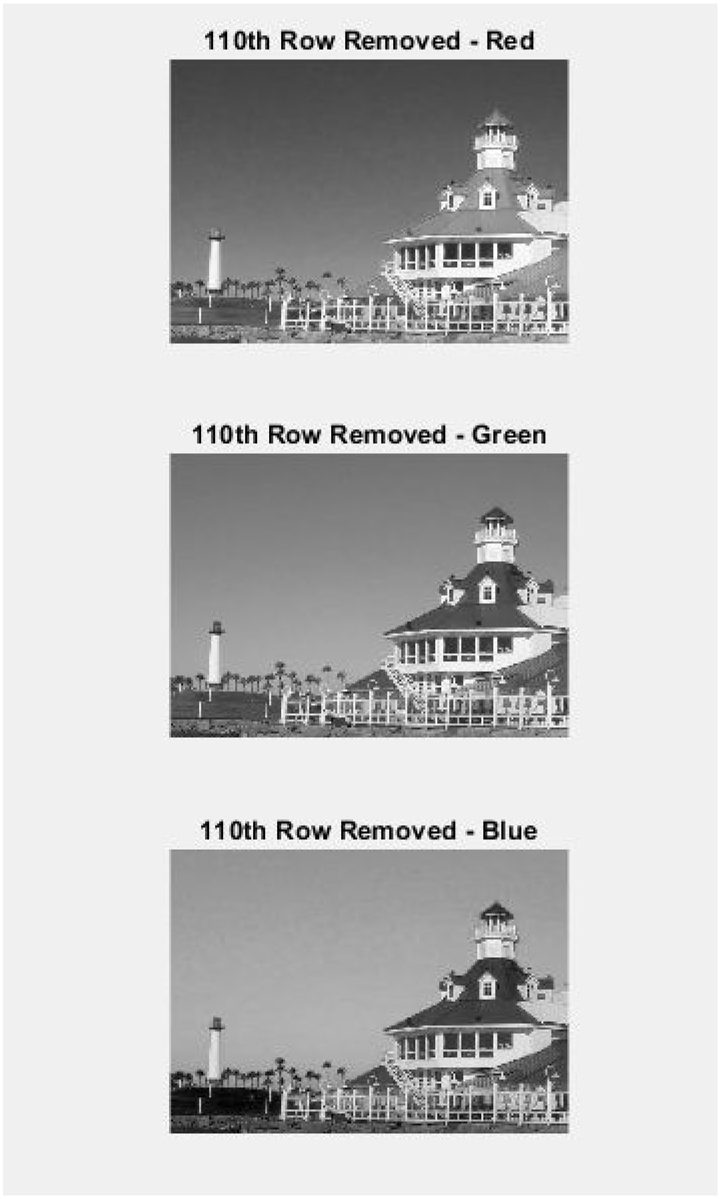


Figure 4: *110th rows removed from R, B, and G formats*

<u>Question G, H</u>

The values of the LLbw2 and LBbw_Red, LBbw_Green, and LBbw_Blue are the same with 357x500 uint8, but all look slightly off in different shades of gray.

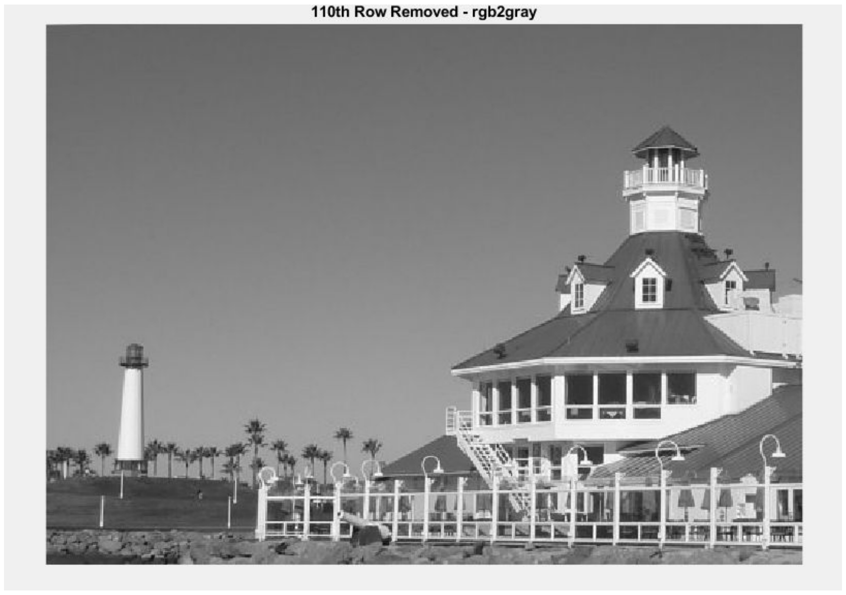


Figure 5: *110th row from grayscale*

<u>Question I, J</u>

The following is a figure of the picture rotated by +30°. And as shown below, in the calculations, picture is not the same size as the original. The original is 358x500 pixels and the new one measures to 560.037x612.013 pixels.

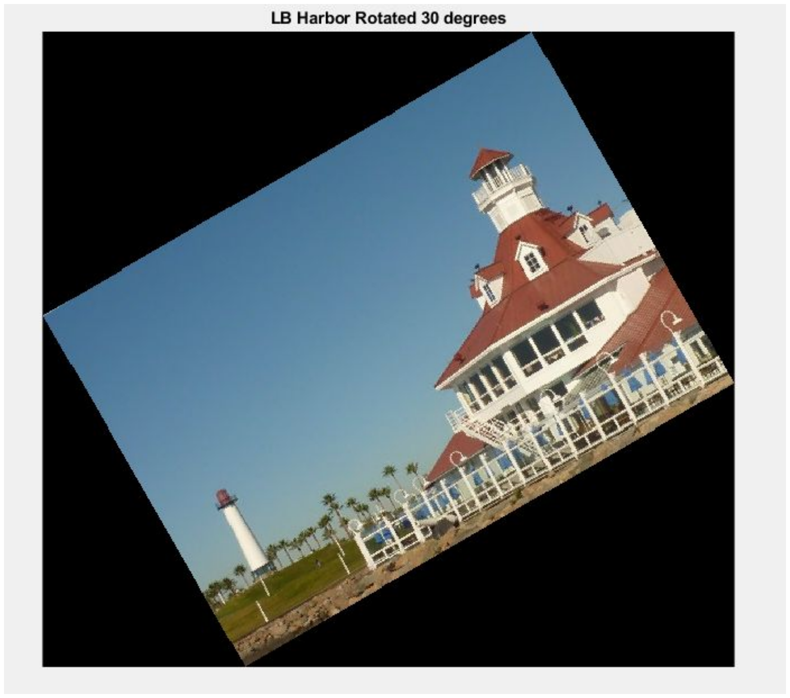


Figure 6: *LB rotated by +30°*

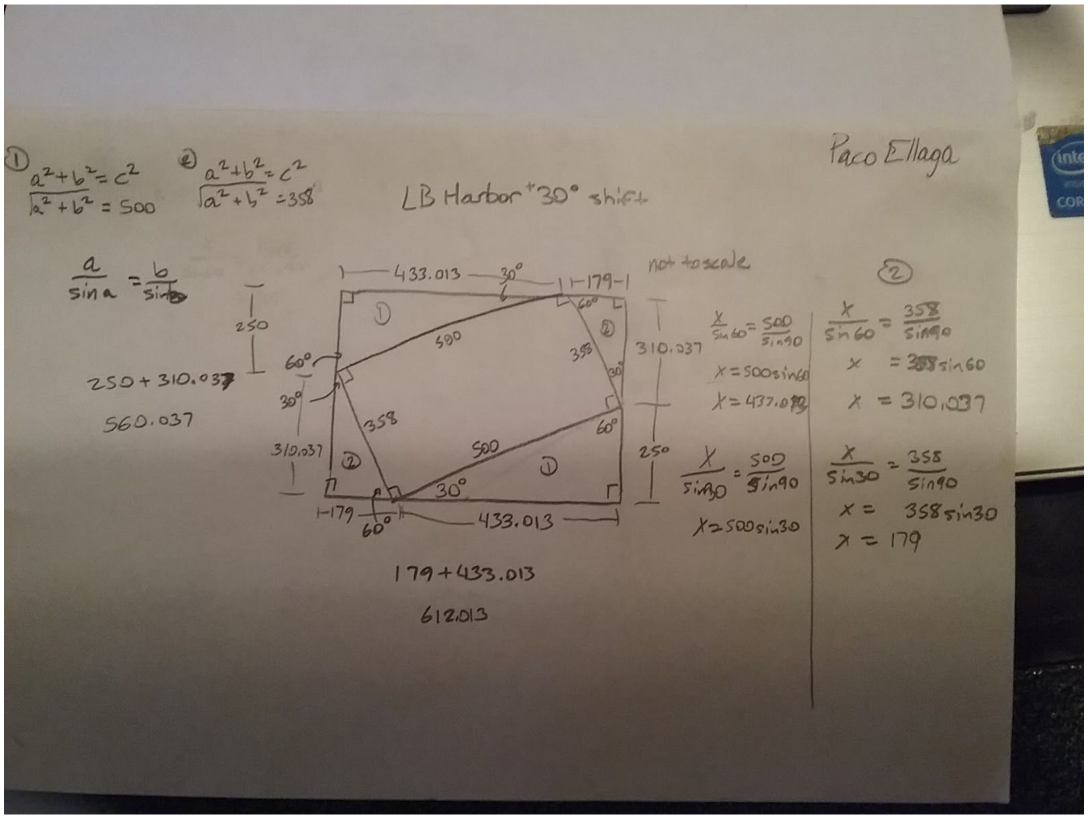


Figure 7: *Calculations of LB rotated by +30°*

Question L, M, N, O

As given, the signal that is needed to be shown is labeled as, xpix = 2*ones(256,1)*cos(2*pi*(0:512)/64). This is shown below as through the *imshow()* command.

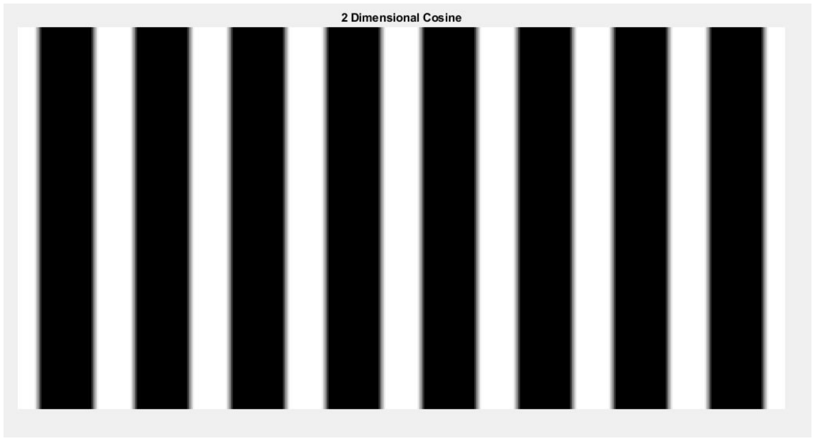


Figure 8: *xpix representation of the cosine signal*

From here, the 30th row was to be removed and displayed as signal. This was done by

xpix30th = xpix(30,:)

As shown below, this is the result of the extraction and the *plot()* of the 30th line of the signal.

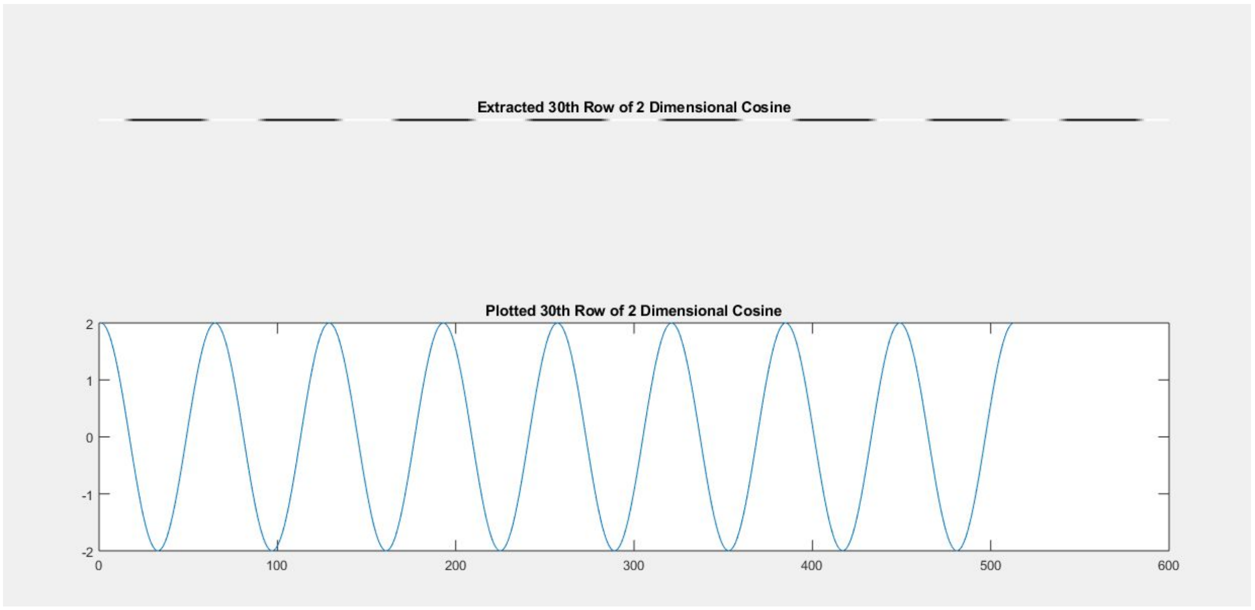
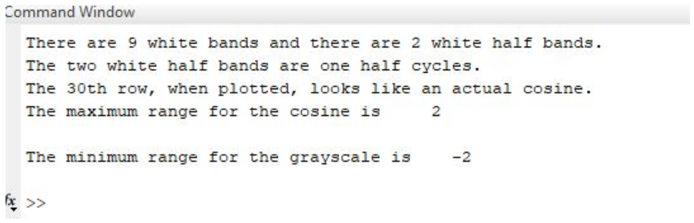


Figure 9: *xpix representation of the cosine signal*

Like the grayscale, the 30th row needs to measured for its maximum and its minimum size. Unfortunately, this changed to be only ranging from -2 to 2. The command to do this shown below:

```
xpix30th_Max = max(max(xpix30th));          % finds max value
fprintf('\nThe maximum range for the cosine is'),disp(xpix30th_Max)

xpix30th_Min = min(min(xpix30th));          % finds min value
fprintf('The minimum range for the grayscale is'),disp(xpix30th_Min)
```



```
There are 9 white bands and there are 2 white half bands.
The two white half bands are one half cycles.
The 30th row, when plotted, looks like an actual cosine.
The maximum range for the cosine is     2

The minimum range for the grayscale is    -2

>>
```

Figure 10: *The max and min range values in the 30throw of the xpix signal*

<u>Question P, Q</u>

In this scenario, xpix_90 is of the 2 dimensional cosine with a phase shift of 90° or π/2. This is displayed as the following signal:

xpix_90 = 2*ones(256,1)*cos(2*pi*(0:512)/64+pi/2);

As shown below, this is the visual representation of the signal compared the original version (without the 90° phase shift). Here, the differences are shown at the edges and signal can be seen to be aligned slightly off from each other.

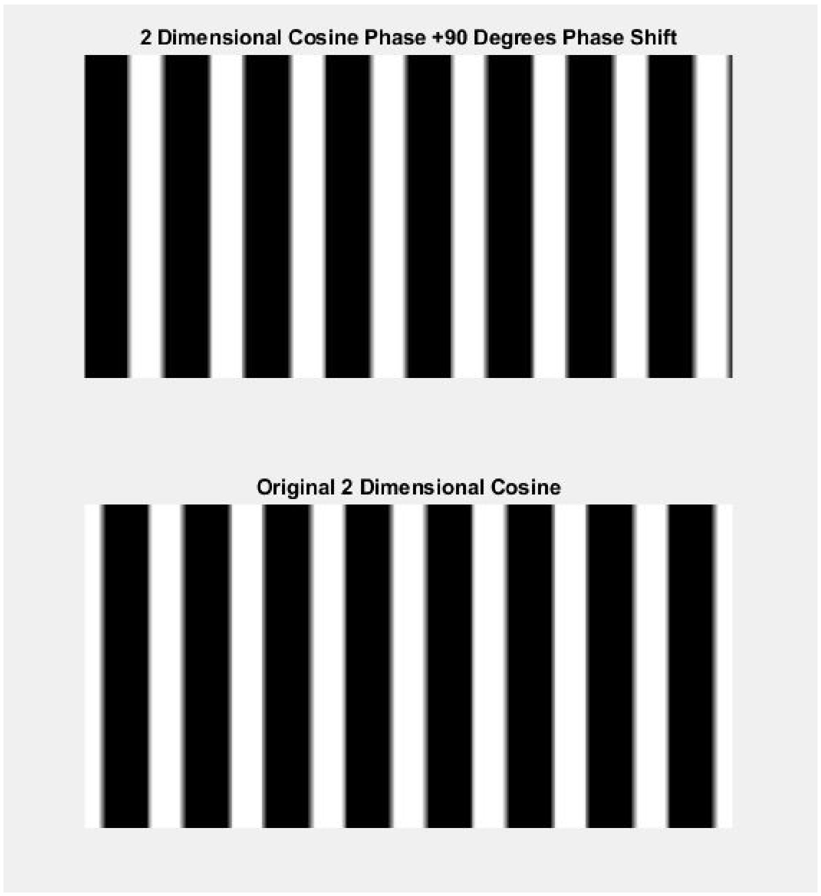


Figure 11: *xpix representation of the cosine signal where the π/2 phase is applied and compared to the original*

Below the is the rotation of the xpix signal and the plot version of it as well.

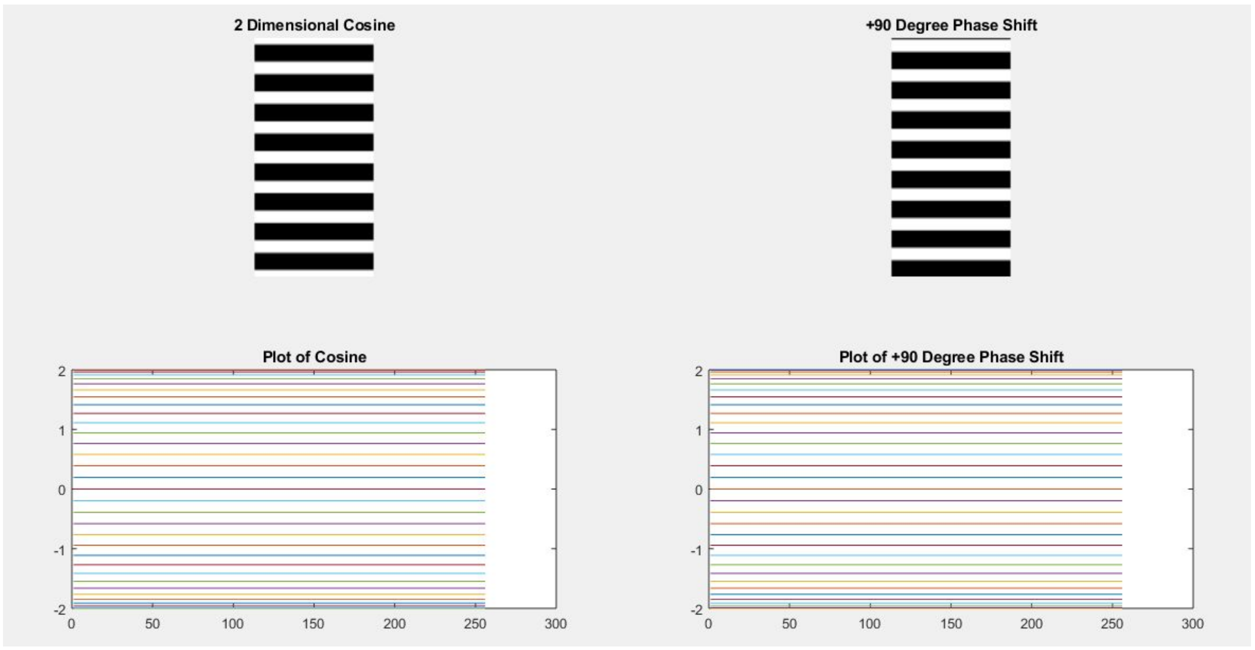


Figure 12: *Rotation and plot of the cosine signal where the π/2 phase is applied and compared to the original*

PART 2

Question A:

As shown in this section downsampling was explored. The image is now at the size of 179x250. In the image, it is apparent that the image noticeably has loss of information on the the lighthouse and the light posts. The following code was used to make the downsampling happen. In the code, the X-axis was down sampled and then the Y-axis downsampled on top of the downsampled X-axis to fully crossover both lateral directions.

```
xDS_LBbw1  = downsample(LBbw,2);
% downsample by a factor of 2 on x-axis
xyDS_LBbw1 = downsample(xDS_LBbw1.',2).';
% downsample by a factor of 2 on y-axis on top of x-axis
```

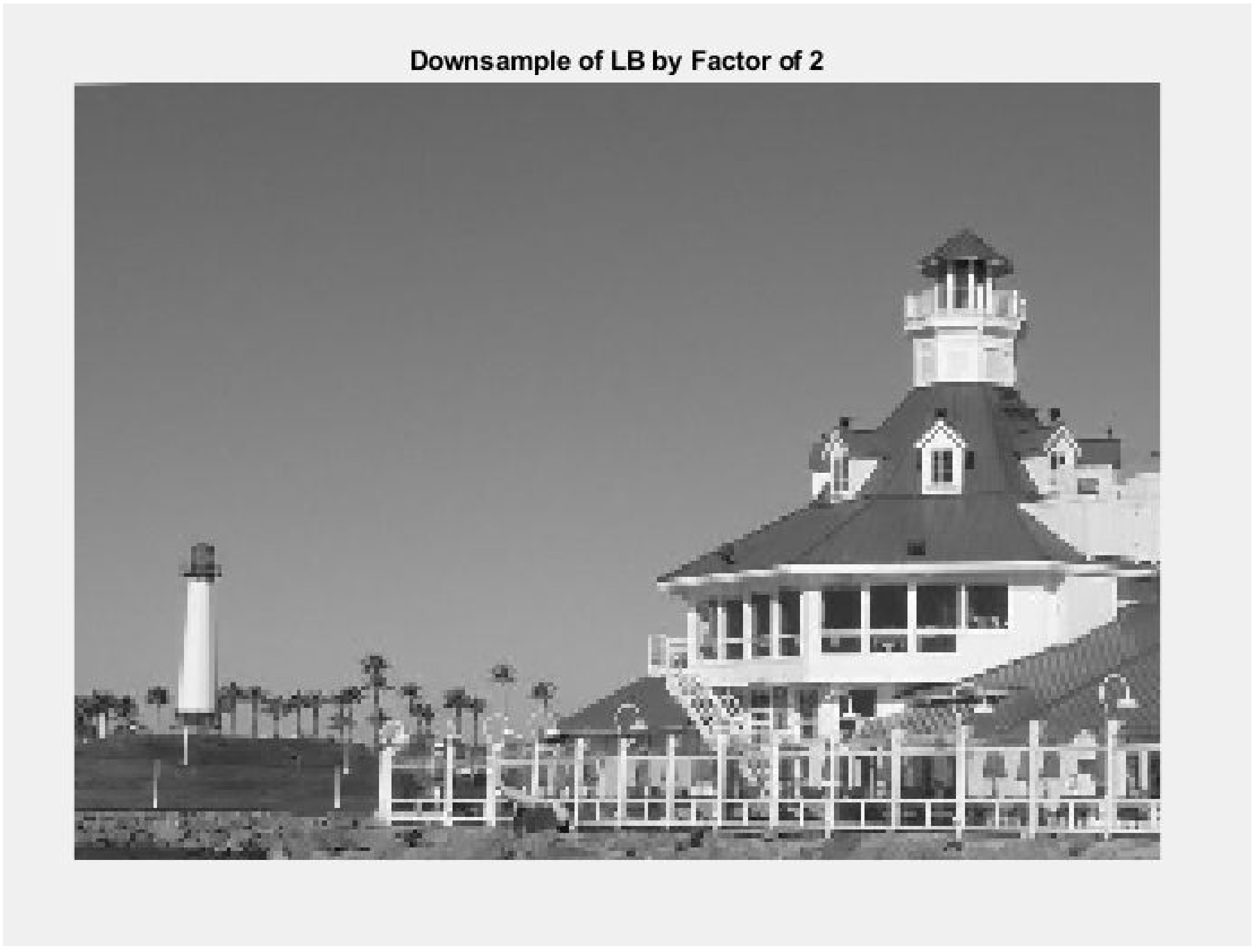


Figure 13: *The downsampled version of the grayscale by a factor of 2*

Question B:

Down sampling creates aliasing by the way it "merges" rows due to reduced amount of pixels. These reduced pixels will squash and make the picture look very "retrographic", similarly seen in old video games and animation. They will present a very distorted look in where you can almost count the amount of pixels being shown on screen. This following picture is a prime example.

Question C:

To reach the limit where the image is not affected by aliasing or the loss of information is the downsampling the process, by downsampling only once, it will not show noticeable changes. This was done the same way as the previous downsample, but the 2 is now a 1 in the parameters.

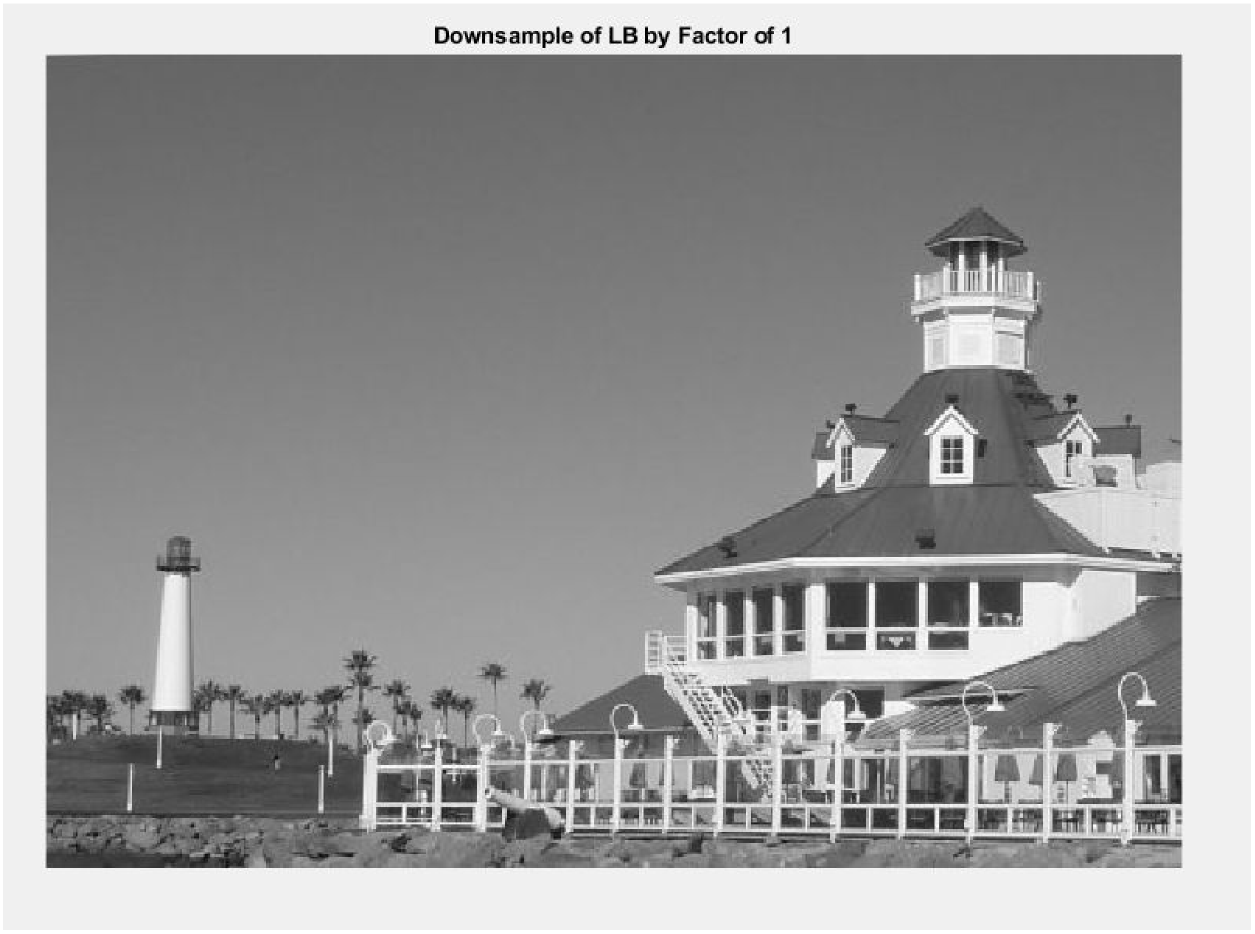


Figure 14: *The downsampled version of the grayscale by a factor of 2*

Part 3

Section: Zero-Order Hold (xr1hold)

Question A

The interpolation factor is 4 in the way that every "section" from positive to negative shows 4 stems. The stem plot is shown after the following code used to generate it:

```
xr1 = (-2).^(0:6);              % signal behavior
L   = length(xr1);              % length based off the signal
nn  = ceil((0.999:1:4*L)/4);    % <-round up to the integer part
xr1hold = xr1(nn);              % holds xr1 to zero order

figure(12)
```

```
stem(xr1hold)
grid minor
title('Stem of Square Pulse Showing "Zero-Order Hold"')
```

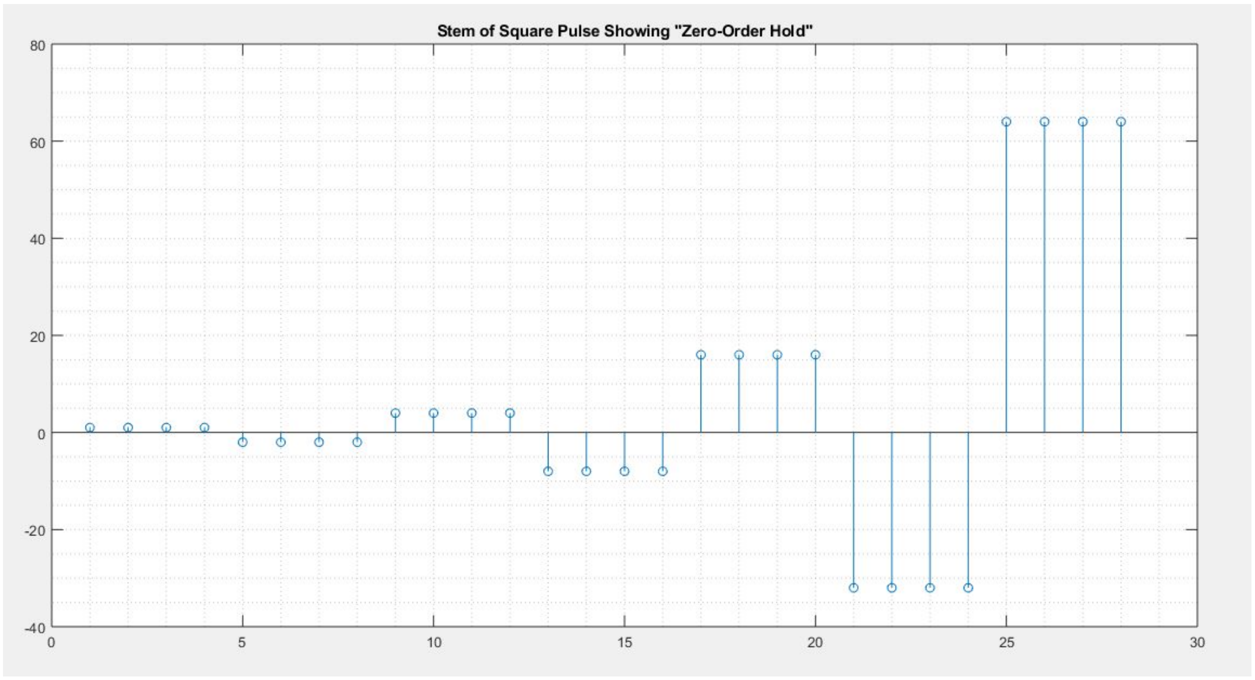


Figure 15: *Stem plot of the zero-order hold after undergoing upsampling*

<u>Question B:</u>

In this section, the upsampling was conducted labeled as xr1hold on the downsampled LBbw -> xyDS_LBbw1. To compare if the process returned it back to its original form. No, the the quality of the image is still a bit distorted, but the values are exactly the same, 179x250 uint8. The uspscale only returned most of its form, but lost some of the values when it was needed to be converted to double and back to uint8. The sample was also not accurate as the downsample was downsampled by 2 and upsampled back up 3. The following code shows how it was done:

```
xr1_LBbw     = size(xyDS_LBbw1);
x_xr1_LBbw   = ceil(0.999:1:3*xr1_LBbw(1)/3);
y_xr1_LBbw   = ceil(0.999:1:3*xr1_LBbw(2)/3);
xr1hold_LBbw = xyDS_LBbw1(x_xr1_LBbw,y_xr1_LBbw);
```

```
figure(13)
subplot(2,1,1)
imshow(xr1hold_LBbw)
title('Zero Order Hold Interpolation by Upsampling')

subplot(2,1,2)
imshow(LBbw2)
title('Original Grayscale before Downsampling')
```

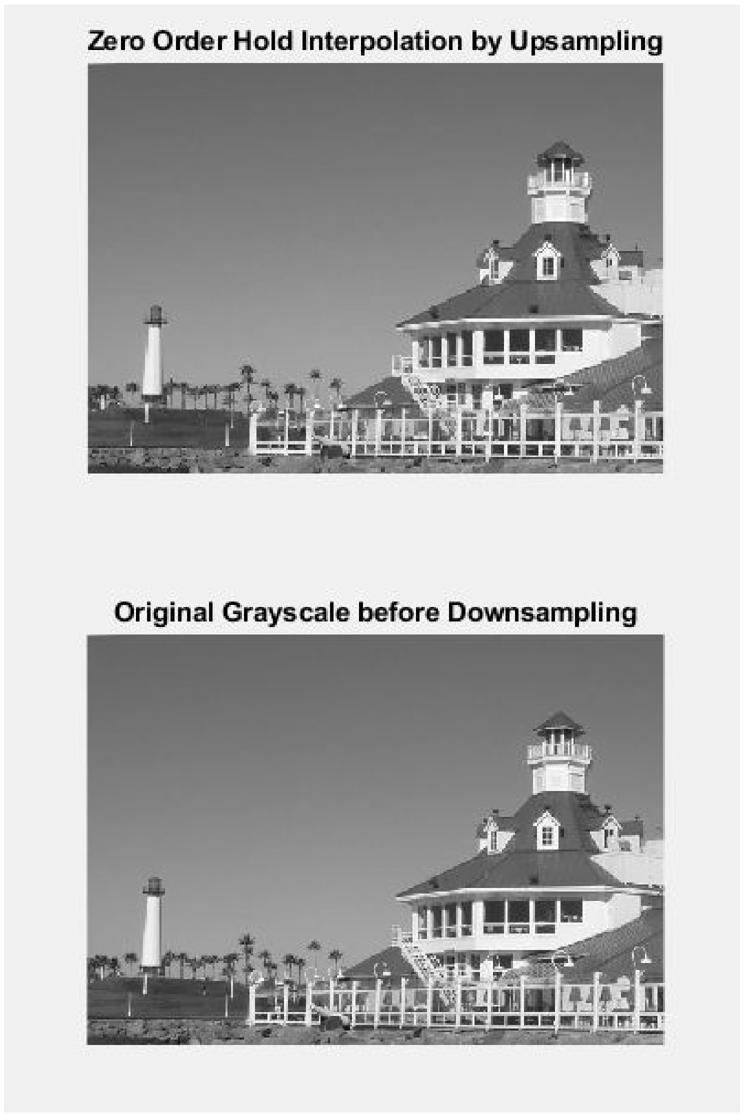

Zero Order Hold Interpolation by Upsampling
Original Grayscale before Downsampling

Figure 16: *xr1hold_LBbw of the zero-order hold after undergoing upsampling factor of 3 from a downsample factor of 2 as it is compared to the original grayscale.*

Section 2: Linear Interpolation (xr1linear)

Question C:

The interpolation factor in the script below is the "n1 = 0:6". This is the length in where it is factored into the signal (xr1), to then be sectioned out in 10ths through "tti". The stem plot is displayed below that and clearly displays its difference between itself and the zero-order hold, where the line of the signal frequency is drawn out versus having square waves.

```
n1       = 0:6;                  % signal length
xr1_L    = (-2).^(n1);            % signal behavior based on length
tti      = 0:0.1:6;               % location between the n1 indices
xr1linear = interp1(n1,xr1_L,tti);  % function is INTERP-ONE

figure(14)
stem(tti,xr1linear)
grid minor
title('Stem of Square Pulse Showing Linear Interpolation')
```

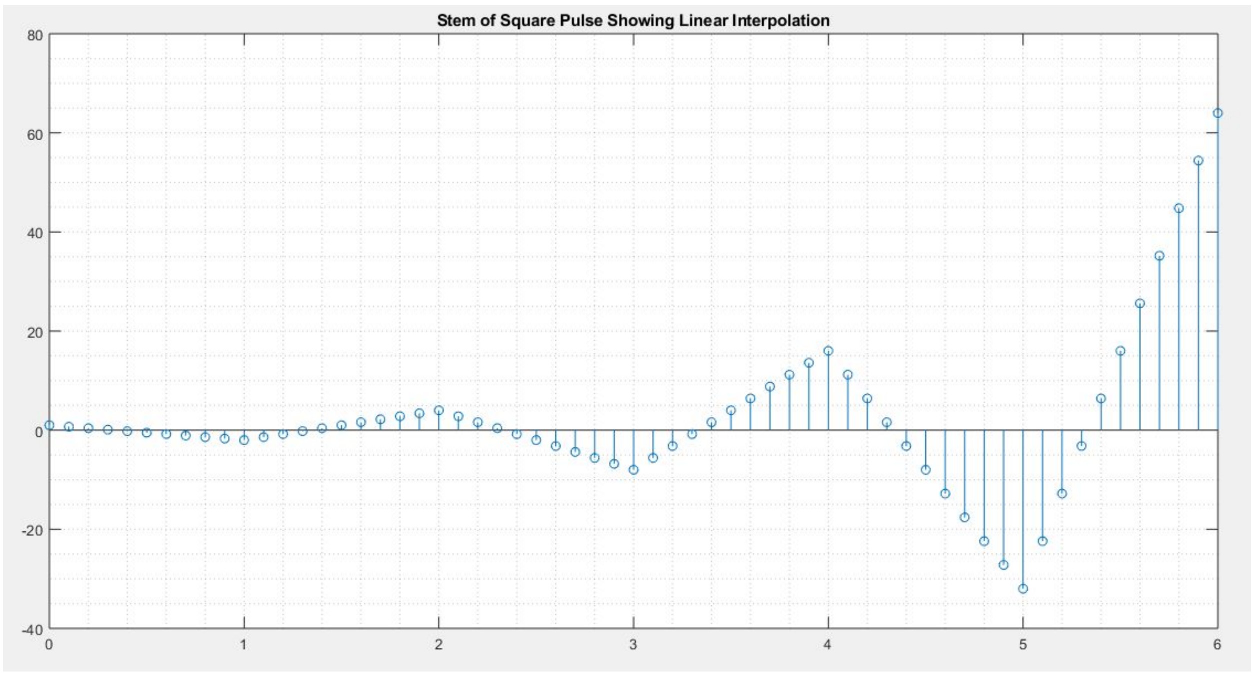


Figure 17: *Stem plot of the linear interpolation after undergoing upsampling*

Question D:

```
%% Up Sampling: xr1linear on Downsampled LBbw -> xyDS_LBbw2
%
% Incomplete section (Here is what I came up with; it doesnt work)

% setting parameters for rescale
% n1_xr1_LBbw2    = [0:255];

% resizing downsample for upsample
% xr1_LBbw2       = size(xyDS_LBbw2);

% x parameters
% x_xr1_LBbw2     = 0:0.1:xr1_LBbw2(1);

% y parameters
% y_xr1_LBbw2     = 0:0.1:xr1_LBbw2(2);

% x with y parameters
% xy_xr1_LBbw2    = xr1_LBbw2(x_xr1_LBbw2,y_xr1_LBbw2);

% sample range
% tti_xr1_LBbw2   = 0:0.1:xr1_LBbw2;

% function for INTERP-ONE
% xr1linear_LBbw2 = interp1(n1_xr1_LBbw2,xy_xr1_LBbw2,tti_xr1_LBbw2);

% figure(15)
% imshow(xr1linear_LBbw2)
% title('Linear Interpolation of xyDS_LBbw')
```

Full MATLAB Code:

```
cdflib.setFileBackward('BACKWARDFILEon');
mode = cdflib.getFileBackward;
% EE386 Project Lab 3
%
% MATLAB code conducted by Paco Ellaga (009602331)
%% Preliminary Work
%
% Double - A type of image where its range values are between [0, 1]. It
% also can be switched into uint8 and vice versa.
%
% Uint8 - A type of image where the range of its capacities stays within
% 0 to 2^8-1 (255) pixels.  It is also short for unsigned 8 bit integer.
%
% Indexed - It is composed of an array(X) and a colormap(mp) that pick the
% red, green, and blue components to create a color.
%
% Binary - With each pixel being dependent on either being a value of 1 or
% 0, the logicalarray it is stored in will create an image while using the
% variable name, "BW".
%
% Intesity - A range of color scaled arrays where double provides [0,1],
% 8-bit provides [0, 255], and 16-bit provides [0 65535].
%
%% Part 1: Manipulating Long Beach Harbor picture (File renamed to "LB")
%
% The size of the picture is 358x500x3
%
% The type is a .jpeg of a uint8.
```

```
who                         % lists current variable (image)
whos

figure(1)
imshow(LB)                  % displays image
title('Long Beach Harbor (LB)')

%% Grayscale Version
%
% The size of the new image is 358x500. When changing to gray the last
% dimension of x3 is removed because it now only in two dimensions.  The x3
% is the dimension representing depth, while the first value (358) is width
% and the second (500) is length.

LBbw = rgb2gray(LB);        % turns image to grayscale
figure(2)
imshow(LBbw)
title('Grayscale of LB')

LBbw_Max = max(max(LBbw));      % finds max value
fprintf('The maximum range for the grayscale is'),disp(LBbw_Max)

LBbw_Min = min(min(LBbw));      % finds min value
fprintf('The minimum range for the grayscale is'),disp(LBbw_Min)

%% Red, Green, and Blue Channelled Versions
%
% The values of all 3 (red, green, blue) are the same with 357x500. But,
% are missing the "x3" at the end like the grayscale. This is because rgb
% relates to [1 1 1] where they are [red green blue]. When removed, the
% channel reflect on the removal. The Red channel has the lightest portion
% in where the red used to be. The green has the removal of the plains and
% the blue has removal of the tint in the sky and the building.

LB_Red      = LB(:,:,1);            % Extracts red channel
LB_Red(110,:) = [];                 % Removes 110th row
figure(3)
subplot(3,1,1)
imshow(LB_Red)
title('110th Row Removed - Red')

LB_Green      = LB(:,:,2);          % Extracts green channel
LB_Green(110,:) = [];               % Removes 110th row
subplot(3,1,2)
imshow(LB_Green)
title('110th Row Removed - Green')

LB_Blue      = LB(:,:,3);           % Extracts blue channel
LB_Blue(110,:) = [];                % Removes 110th row
subplot(3,1,3)
imshow(LB_Blue)
title('110th Row Removed - Blue')
%% Grayscale Comparison
%
% The values of the LLbw2 and LBbw_Red, LBbw_Green, and LBbw_Blue are the
% same with 357x500 uint8, but all look slightly off in different
% shades of gray.
```

```
LBbw2        = rgb2gray(LB);            % creates new grayscale
LBbw2(110,:) = [];                      % removes 110th row
figure(4)
imshow(LBbw2)
title('110th Row Removed - rgb2gray')

%% 30 Degrees Rotation Version
%
% The following is a figure of the picture rotated by +30 degrees. And as
% shown below, in the calculations, picture is not the same size as the
% original. The original is 358x500 pixels and the new one measures to
% 560.037x612.013 pixels.

figure(5)
imshow(imrotate(LB,30))                 % rotates image by +30 degrees
title('LB Harbor Rotated 30 degrees')

%% Creating a 2 Dimensional Cosine
%

xpix = 2*ones(256,1)*cos(2*pi*(0:512)/64);

figure(6)
imshow(xpix)
title('2 Dimensional Cosine')
fprintf('There are 9 white bands and there are 2 white half bands. \nThe two white half bands are
one half cycles.')
```

```
xpix30th = xpix(30,:);                    % Removes 30th row

figure(7)
subplot(2,1,1)
imshow(xpix30th)
title('Extracted 30th Row of 2 Dimensional Cosine')

subplot(2,1,2)
plot(xpix30th)
title('Plotted 30th Row of 2 Dimensional Cosine')
fprintf('\nThe 30th row, when plotted, looks like an actual cosine.')

xpix30th_Max = max(max(xpix30th));          % finds max value
fprintf('\nThe maximum range for the cosine is'),disp(xpix30th_Max)

xpix30th_Min = min(min(xpix30th));          % finds min value
fprintf('The minimum range for the grayscale is'),disp(xpix30th_Min)

%% Manipulating the 2 Dimensional Cosine
%
% xpix_90 is of the 2 dimensional cosine with a phase shift of 90 degrees.

xpix_90 = 2*ones(256,1)*cos(2*pi*(0:512)/64+pi/2);

figure(8)
subplot(2,1,1)
imshow(xpix_90)
title('2 Dimensional Cosine Phase +90 Degrees Phase Shift')
```

```
subplot(2,1,2)
imshow(xpix)
title('Original 2 Dimensional Cosine')
fprintf('\nThe new signal is shift over to the left so the signal starts at the black bar now. \nThis is
the 90 degree phase error showing in the signal.')

figure(9)
subplot(2,2,1)
imshow(imrotate(xpix,90))       % +90 degree rotation of 1st cosine
title('2 Dimensional Cosine')

subplot(2,2,2)
imshow(imrotate(xpix_90,90))    % +90 degree rotation of 2nd cosine
title('+90 Degree Phase Shift')

subplot(2,2,3)
plot(xpix)
title('Plot of Cosine')

subplot(2,2,4)
plot(xpix_90)
title('Plot of +90 Degree Phase Shift')

%% Part 2: Down Sampling
%
% Minimum Spatial Frequency for the First Lightpost
xDS_LBbw1  = downsample(LBbw,2);
% downsample by a factor of 2 on x-axis
xyDS_LBbw1 = downsample(xDS_LBbw1.',2).';
```

```
% downsample by a factor of 2 on y-axis on top of x-axis

figure(10)
imshow(xyDS_LBbw1)
title('Downsample of LB by Factor of 2')
fprintf('Leftmost lightpost starts aliasing at a downsample factor of 2.')

% Minimum Spatial Frequency for no aliasing
xDS_LBbw2  = downsample(LBbw,1);
% downsample by a factor of 1 on x-axis
xyDS_LBbw2 = downsample(xDS_LBbw2.',1).';
% downsample by a factor of 1 on y-axis on top of x-axis

figure(11)
imshow(xyDS_LBbw2)
title('Downsample of LB by Factor of 1')
fprintf('\nThe minumum for avoiding aliasing at a downsample factor of 1.')

%% Explanation of Downsampling
%
% Down sampling creates aliasing by the way it "merges" rows due to reduced
% amount of pixels. These reduced pixels will squash and make the picture
% look very "retrographic", similarly seen in old video games and
% animation. They will present a very distorted look in where you can
% almost count the amount of pixels being shown on screen. This following
% picture is a prime example.

%% Part 3: Up Sampling: Square Pulse to Zero-Order Hold (xr1hold)
%
```

```
% The interpolation factor is 4 in the way that every "section" from
% positive to negative shows 4 stems.

xr1 = (-2).^(0:6);              % signal behavior
L   = length(xr1);              % length based off the signal
nn  = ceil((0.999:1:4*L)/4);    % <-round up to the integer part
xr1hold = xr1(nn);              % holds xr1 to zero order

figure(12)
stem(xr1hold)
grid minor
title('Stem of Square Pulse Showing "Zero-Order Hold"')

%% Up Sampling: xr1hold on Downsampled LBbw -> xyDS_LBbw1
%
% No, the the quality of the image is still a bit distorted, but the
% values are exactly the same, 179x250 uint8. The uspscale only returned
% it's form, but lost some of the values when it was needed to be converted
% to double and back to uint8.

xr1_LBbw     = size(xyDS_LBbw1);
x_xr1_LBbw   = ceil(0.999:1:3*xr1_LBbw(1)/3);
y_xr1_LBbw   = ceil(0.999:1:3*xr1_LBbw(2)/3);
xr1hold_LBbw = xyDS_LBbw1(x_xr1_LBbw,y_xr1_LBbw);

figure(13)
subplot(2,1,1)
imshow(xr1hold_LBbw)
title('Zero Order Hold Interpolation by Upsampling')
```

```
subplot(2,1,2)
imshow(LBbw2)
title('Original Grayscale before Downsampling')

%% Up Sampling: Linear Interpolation (xr1linear)
%
% The interpolation factor is the "n1 = 0:6". This is the length in where
% it is factored into the signal (xr1), to then be sectioned out in 10ths
% through "tti".

n1        = 0:6;                    % signal length
xr1_L     = (-2).^(n1);             % signal behavior based on length
tti       = 0:0.1:6;                % location between the n1 indices
xr1linear = interp1(n1,xr1_L,tti);  % function is INTERP-ONE

figure(14)
stem(tti,xr1linear)
grid minor
title('Stem of Square Pulse Showing Linear Interpolation')

%% Up Sampling: xr1linear on Downsampled LBbw -> xyDS_LBbw2
%
% Incomplete section (Here is what I came up with; it doesnt work)

% setting parameters for rescale
% n1_xr1_LBbw2    = [0:255];

% resizing downsample for upsample
% xr1_LBbw2       = size(xyDS_LBbw2);

% x parameters
% x_xr1_LBbw2     = 0:0.1:xr1_LBbw2(1);

% y parameters
% y_xr1_LBbw2     = 0:0.1:xr1_LBbw2(2);

% x with y parameters
% xy_xr1_LBbw2    = xr1_LBbw2(x_xr1_LBbw2,y_xr1_LBbw2);

% sample range
% tti_xr1_LBbw2   = 0:0.1:xr1_LBbw2;

% function for INTERP-ONE
% xr1linear_LBbw2 = interp1(n1_xr1_LBbw2,xy_xr1_LBbw2,tti_xr1_LBbw2);

% figure(15)
% imshow(xr1linear_LBbw2)
% title('Linear Interpolation of xyDS_LBbw')
```