

**Lab 3: Image as a Discrete 2D Signal**

By. Christopher W. Fingers

010099178

**Preliminary Work:**

Double: A large value of floating-point integers that can utilizes a larger size of integers.

Unit8: This is an 8-bit unsigned integer that takes a range of pixels and assigns them accordingly into an array.

Indexed: An array and a color map matrix that uses pixel values to map the colormaped values. The image has a set of 3 different color arrays ranging from red, green and blue that when overlapped can produce the desired color for the image.

Binary: An image that is generally represented in grey scale, as in a 1 for white and a 0 for black in terms of the color array associated with the image.

Intensity: This image has a range from 0 to 1 of color represented in a black and white scale. Instead of binary, there is a spectrum of grey that can represent the shade of a color.

**Part 1:**

a. The image is 358x500x3, which has 537,000 elements. The image is a unit8 type.

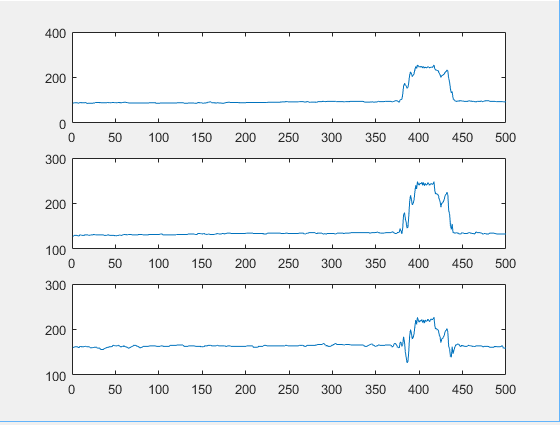
****

**Figure 1: Grey Scaled Long Beach Harbor.**

d. The grey version of the Long Beach harbor is 358x500 units, which means it has 179,000

elements. The reason for the single vector is because the previous colored picture has 3 colors that can be combined to make up the colors in the image: red, green and blue. Varying frequencies between these colors affect how intense a certain color is at a certain pixel.

This is why when the picture is converted to grey there are three different signals missing from it.

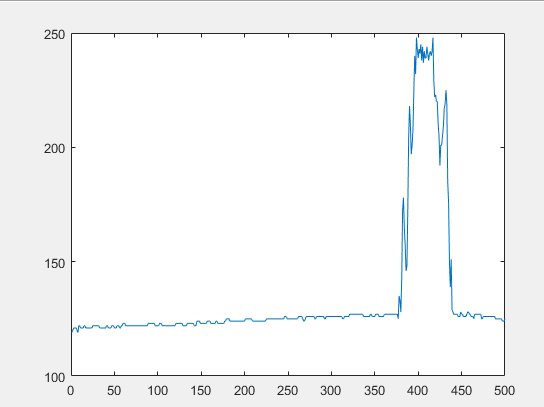
****

**Figure 2: Red, Green, Blue frequencies at row 110.**

f. The three subplots show 3 varying signals with different starting amplitudes that all spike between pixels 350 to 450. These three signals represent the intensity of red, green and blue on the image. Blue has a relatively higher frequency throughout the picture until it

comes in contact with the building. It still spikes, but not as much as the others. Red and green on the other hand stay relatively low in frequency until the pixel range stated above. This is more than likely because the building itself needs to be colored around the pixel range

350 - 450 as based on by the picture. The portion that is being colored seems to be the very top of the building.

****

**Figure 3: Grey Scale row 110 frequency.**

h. The grey scaled figure has a similar behavior to the rgb frequency, except with a signal value. The intensity of the grey scaled figure resembles the blue signals frequency for the colored picture. This makes sense because the intensity of the grey scaled image should stay relatively the same as the rgb image with the high intensity.

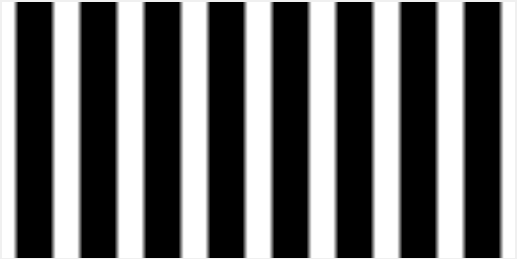
****

**Figure 4: Rotated image of the Long Beach Harbor by 30 degrees**

**j.**

Because of the angle shift of the photo the scaling between the row and columns for the pixels needs to be altered to fit the rotated picture. The overall size of the picture can be calculated by using sin and cos of 30 degrees and multiplying by the original row and column’s length value. This is because the original images row and column’s size value is related to the hypotenuse of the new pictures empty space caused by the 30 degree shift. When these values are found by adding together the new empty space triangles height and width appropriately, the new size of the image can be found.

k. Based on max and min the low value is 11 and the high value is 255.

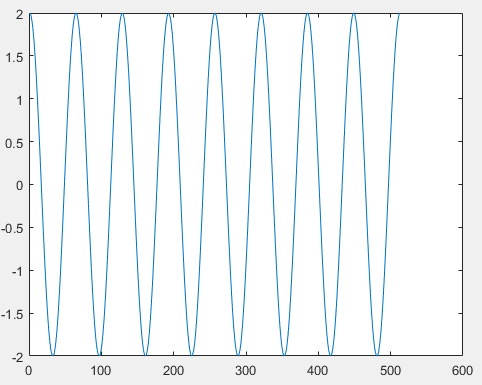


**Figure 5: Half white Half black bands**

l. There are 6 full white bands and 2 half bands. The half bands are caused because of the cosine function. The cosine function is an even function, which means that the amplitude of the positve x-axis of the cosine function will be identical to the negative x-axis of the cosine function. The half white band at the beginning of the bands has the other half of

the white if the band signal was continued to the negative.

m. The width of each band is related to the number of positive amplitudes are plotted versus the number of negative amplitudes are plotted.



**Figure 6: Row 30 of the half white and half black band cosine signal.**

n/o. It looks like a cosine function with a max and min amplitude of 2 and -2 respectively.



**Figure 7: Shifted pi/2 signal.**

The image was phase shifted to the right by pi/2. This causes the rest of the image to shift to the right, showing the rest of the white band.

****

**Figure 8: Vertical Half black Half white bands shifted by pi/2.**

**Part 1 MatLab Code:**

% Lab 3 Part 1

who longBeachHarbor

% The image is 358x500x3. It has 537,000 elements associated with it.

% The image is a unit8 type.

figure

LBbw=rgb2gray(longBeachHarbor);

imshow(LBbw)

% The grey version of the Long Beach harbor is 358x500 units, which means

% it has 179,000 elements. The reason for the single vector is because the

% previous colored picture has 3 colors that can be combined to make up the

% colors in the image: red, green and blue. Varying frequencies between

% these colors affect how intense a certain color is at a certain pixel.

% This is why when the picture is converted to grey there are three

% different signals missing from it.

figure

imwrite(LBbw,'LBbw.png');

subplot(3,1,1);

plot(longBeachHarbor(110,1:500,1))

subplot(3,1,2);

plot(longBeachHarbor(110,1:500,2))

subplot(3,1,3);

plot(longBeachHarbor(110,1:500,3))

% The three subplots show 3 varying signals with different starting

% amplitudes that all spike between pixels 350 to 450. These three

% signals represent the intensity of red, green and blue on the image.

% Blue has a relatively higher frequency throughout the picture until it

% comes in contact with the building. It still spikes, but not as much as

% the others. Red and green on the other hand stay relatively low in

% frequency until the pixel range stated above. This is more than likely

% because the building itself needs to be colored around the pixel range

% 350 - 450 as based on by the picture. The portion that is being colored

% seems to be the very top of the building.

figure

plot(LBbw(110,1:500));

% The greyscaled figure has a similar behavior to the rgb frequency, except

% with a signal value. The intensity of the greyscaled figure resembles

% the blue signals frequency for the colored picture. This makes sense

% because the intensity of the greyscaled image should stay relatively the

% same as the rgb image with the high intensity.

%%

figure

x=imrotate(longBeachHarbor,30);

imshow(x)

highValue=max(max(LBbw))

lowValue=min(min(LBbw))

%%

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

imshow(xpix)

% There are 6 full white bands and 2 half bands. The half bands are caused

% because of the cosine function. The cosine function is an even function,

% which means that the amplitude of the positve x-axis of the cosine

% function will be identical to the negative x-axis of the cosine function.

% The half white band at the beginning of the bands has the other half of

% the white if the band signal was continued to the negative.

%%

figure

plot(xpix(30,1:513))

highValue2=max(max(xpix))

lowValue2=min(min(xpix))

% It looks like a cosine function with a max and min amplitude of 2 and -2

% respectively.

%%

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

imshow(xpix)

%The image was phase shifted to the right by pi/2. This causes the rest of

%the image to shift to the right, showing the rest of the white band.

%%

figure

x=imrotate(xpix,90);

imshow(x)

**Part 2:**

**a.**

****

**Figure 9: Downscaled Long Beach Harbor.**

The new image is 179x250, which means it has 44,750 elements in it’s matrix.

**b.**

****

**Figure 10: Zoom in of the light house.**

When aliasing occurs with higher frequencies the amplitude of the frequency starts to bounce back into a lower frequency area. This is because of sampling issues that causes an image’s spectra to not display an image properly. Looking at the palm tree, the top and side of the lighthouse, the fence in front and the hidden canon in front of the fence on the right. The pixilation of these spots seems to blend together into a blur.

****

**Figure 11: Zoom in of the building.**

The stairs of the building, the fence in front, the gazebo on the top right of the picture and the light posts in front of the build seemingly blend together with the background and images around it. A clear sign of aliasing based on down scaling.

c. The minimum spatial frequency for the light post is 3 Hz.

**Part 2 MatLab Code:**

%% EE 386 Lab 3 part 2

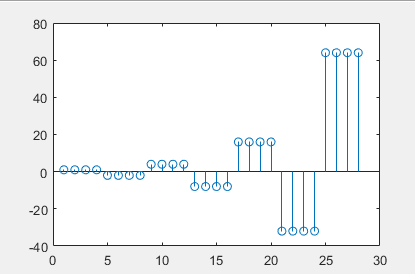
%Downscaling saves the first row and columnn value, then skips the next two

%sample values and saves the following value.

DownScaleLBbw=LBbw(1:2:358, 1:2:500);

imshow(DownScaleLBbw)

**Part 3:**

****

**Figure 12: Zero Order Hold Example.**

1. The interpolation factor in this case is 4.

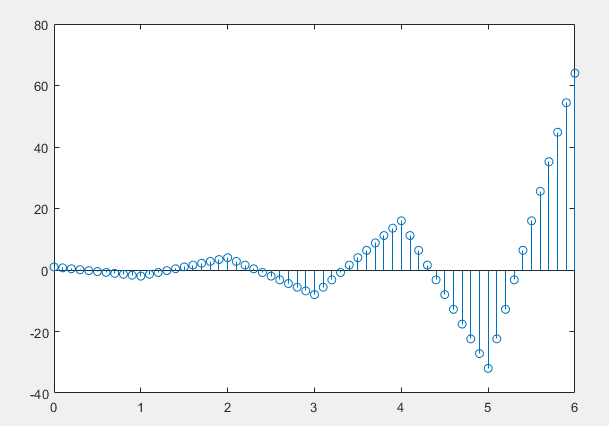


**Figure 13: Zero Order Hold of downscaled image.**



**Figure 14: Original Grey Scaled Long Beach Harbor.**

1. When comparing the original image to the zero-order hold image there is some issues with the reconstruction. There is still pixilation in areas such as the palm trees, edges of most objects and the light post. This makes sense because the zero-order hold fills in the missing elements in the matrix with values that are already associated with the image. A transition from the sky to the building or small objects will still cause pixilation because the original values cannot be replaced, however the graininess of the image is greatly reduced. Objects that were previously nearly impossible to see such as the cannon and the light post directly in front of the building are easier to see now.

****

**Figure 15: Linear Interpolation example.**

1. The interpolation factor when converting is 10.



**Figure 16: Linear Interpolation of Long Beach Harbor**

1. Unlike the zero-order hold method, the linear interpolation method does not have as many problems with edges when reconstructing the image. The image is fuzzier in certain spots, such as the middle of the building, however the most obvious difference is the pixilation on the palm trees and light post is much better than the zero-order hold. In my personal opinion the linear interpolation seems to produce a more detailed image than the zero-order hold, especially on small or thing objects, which I think is much better for image reconstruction.

**Part 3 MatLab Code:**

%% EE 386 Lab3 Part 3

xr1=(-2).^(0:6);

L=length(xr1);

nn=ceil((0.999:1:4\*L)/4);

xr1hold=xr1(nn);

stem(xr1hold)

%% Zero-Order Hold

xn = ceil((0.999:1:3\*length(DownScaleLBbw(:,1)))/3);

yn = ceil((0.999:1:3\*length(DownScaleLBbw(1,:)))/3);

figure

rescaled = DownScaleLBbw(xn, yn);

figure

imshow(rescaled)

figure

imshow(LBbw)

%% Linear

n1= 0:6;

xr1=(-2).^n1;

tti=0:0.1:6;

xr1linear=interp1(n1,xr1,tti);

stem(tti,xr1linear);

%%

ddyt=DownScaleLBbw';

dd1=double(DownScaleLBbw);

dd2=double(ddyt);

t = 1:0.4:179;

t2= 1:0.4:250;

x1=length(dd1(:,1));

y1=length(dd1(1,:));

xlinear = interp1(1:179,dd1,t);

ddy = xlinear.';

ylinear = interp1(1:250,ddy,t2);

LinearRecon=ylinear';

imshow(mat2gray(LinearRecon))