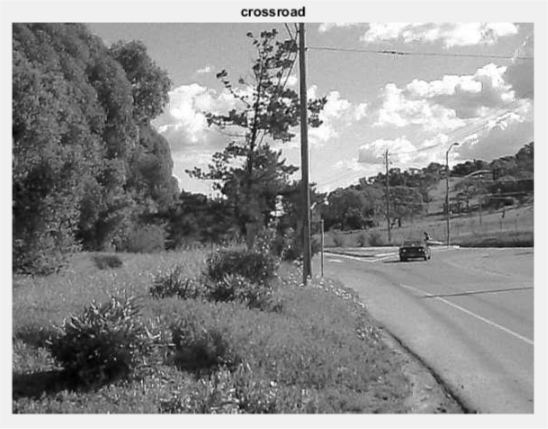
**Digital Image Processing Sessional Assignment 1**

**150222**

**Exercise 1**

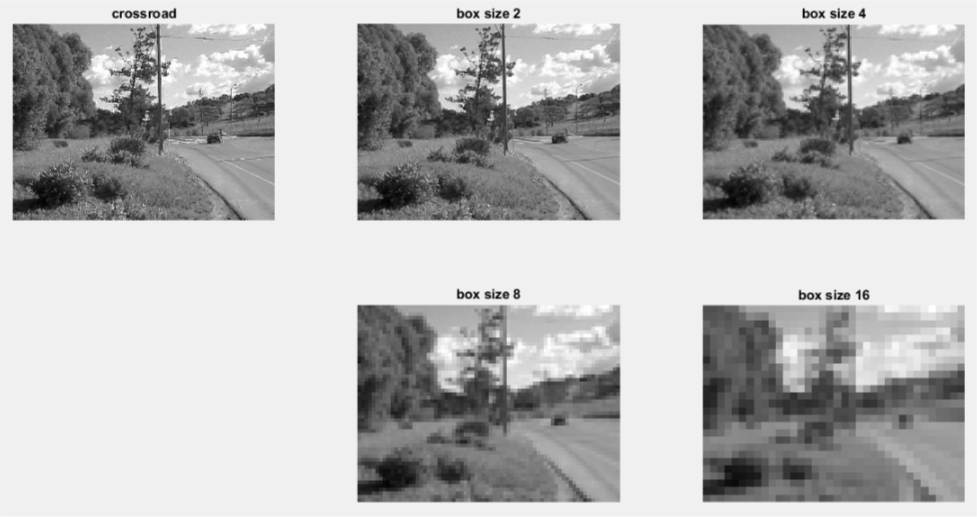
This problem was to understand image resolution.

1. For this problem it was told to Download the provided “crossroad.dat” file. After that import it with fopen() and then read it with fread() then assigned data of unsigned 8 bits while having width of 580 pixels and height of 435 pixels. Finally the image was shown with imshow().



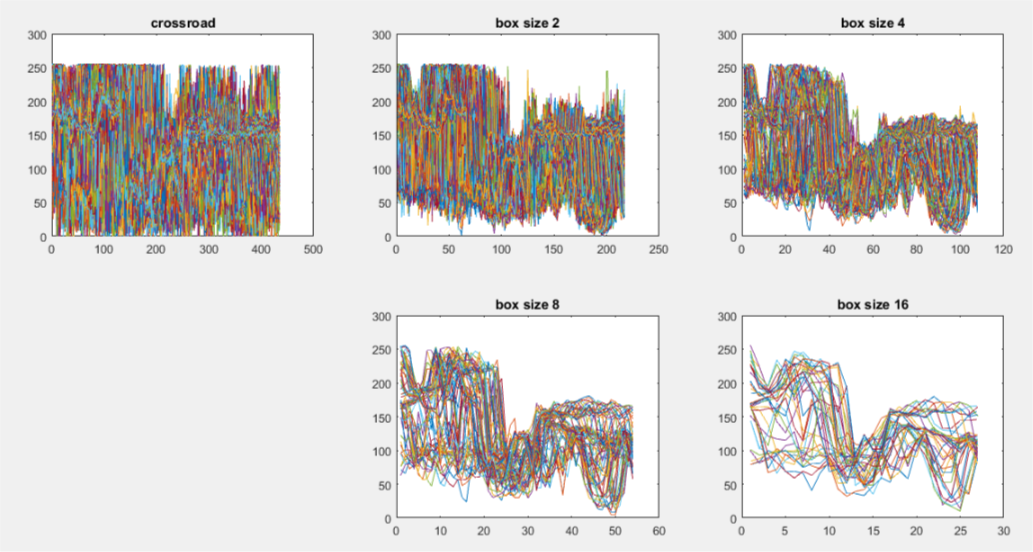
As the image was not suitably oriented so it was rotate using imrotate(). Thus, the observation for this problem was done.

1. For this part of the problem it was told to take the image used in the previous part and change the image resolution with boxes 2,4,8 1nd 16 pixels meaning decreasing the pixel numbers by .5,.25,.125,0.0625 of the original image. For that purpose the main image was resize by using imresize() having new heights and widths divided by 2,4,8 and 16 .



By seeing the resulted images, it is clear that by reducing pixel numbers the image loses its resolution. By applying box with 2x2 pixels only the sharpness is reduced without it no major difference is seen. For 4x4 the image gets blurry and the pixel boxes are seen at some parts. For 8x8 all over the image is the pixel boxes are seen and the image is more blur and the sharp edges of the pixel boxes giving the image a box effect. For 16x16 the pixel boxes are more bigger and the edges of the boxes are seen more and the image is holding the lowest resolution and details. So, in the conclusion it can be said that the more pixels are used to form box the more the image loses its resolution.

1. This part told to examine minimum obvious structure size in each case. For that the images were plot and graph showed their obvious structure.



The observation from the plots are that the main image holds more information as it has more pixels by decreasing of pixels the images lost their information and so that their obvious structure is getting shrunk.

**Code:**

% importing the .dat file

I = fopen('crossroad.dat');

% converting the .dat file to matlab readable image format

I = fread(I,[580,435]);

% rotating the image to get a nice view

I = imrotate(I,270);

%converting the image into uint8 format

I = uint8(I);

% displaying the image

subplot(2,3,1),imshow(I),title('crossroad');

% Changing the image resolution sizes with 2,4,8,16 boxes

w = floor(580/2);

h = floor(435/2);

I2 = imresize(I,[h w]);

subplot(2,3,2),imshow(I2),title('box size 2');

w = floor(580/4);

h = floor(435/4);

I4 = imresize(I,[h w]);

subplot(2,3,3),imshow(I4),title('box size 4');

w = floor(580/8);

h = floor(435/8);

I8 = imresize(I,[h w]);

subplot(2,3,5),imshow(I8),title('box size 8');

w = floor(580/16);

h = floor(435/16);

I16 = imresize(I,[h w]);

subplot(2,3,6),imshow(I16),title('box size 16');

% Displaying the obvious structure

figure;

subplot(2,3,1), plot(I), title('crossroad');

subplot(2,3,2), plot(I2), title('box size 2');

subplot(2,3,3), plot(I4), title('box size 4');

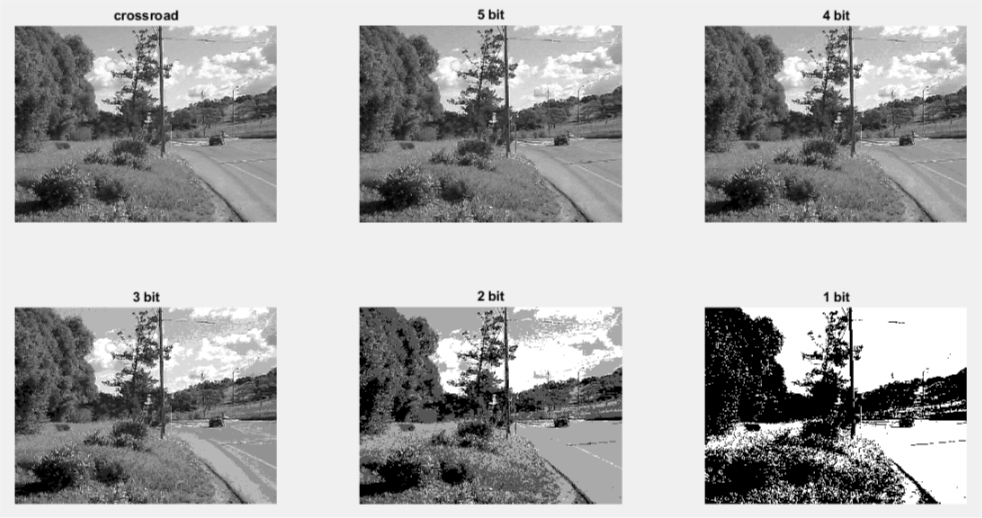
subplot(2,3,5), plot(I8), title('box size 8');

subplot(2,3,6), plot(I16), title('box size 16');

**Exercise 2**

This problem told to understand quantization which is intensity resolution by experimentation which can be done by reducing the data of each pixel.

1. This told to read the previous image which was done by the same process but assigning it to unsigned 8-bit image was not done as for the next part the pixel values were needed to divide.
2. The initial image holds 8-bit data for each pixel but now it was told to change it to 5-bit,4-bit,3-bit,2-bit,1-bit data. For doing that, we needed to divide all the pixels as 5-bit means there will be 25 = 32 meaning that 0 to 31 per pixels. So that all the pixels of the main image were divided by 8. Now by the same process pixels were divided by 16,32,64 and 128 for assigning 4,3,2,1-bit value per pixels. As dividing makes the image as matrix so it was then converted to gray using mat2gray().

****

From the result it is seen that the less the pixel value the less details it holds and for the 1-bit image it turns into a binary image basically. By observing the images, it can be said that the 5-bit image preserves more data than others. And it does change from place to place in an image as there are different pixel values and dividing them changes their values rapidly. As example we can say that for the 1-bit image while dividing with 128 the values goes either to 1 or 0 as the floor value is taken.

**Code:**

% importing the .dat file

I = fopen('crossroad.dat');

% converting the .dat file to matlab readable image format

I = fread(I,[580,435]);

% rotating the image to get a nice view

I = imrotate(I,270);

% displaying the image

subplot(2,3,1),imshow(mat2gray(I)),title('crossroad');

% applying 5 bit data for each pixel

I5 = floor(I/8);

subplot(2,3,2),imshow(mat2gray(I5)),title('5 bit');

% applying 4 bit data for each pixel

I4 = floor(I/16);

subplot(2,3,3),imshow(mat2gray(I4)),title('4 bit');

% applying 3 bit data for each pixel

I3 = floor(I/32);

subplot(2,3,4),imshow(mat2gray(I3)),title('3 bit');

% applying 2 bit data for each pixel

I2 = floor(I/64);

subplot(2,3,5),imshow(mat2gray(I2)),title('2 bit');

% applying 1 bit data for each pixel

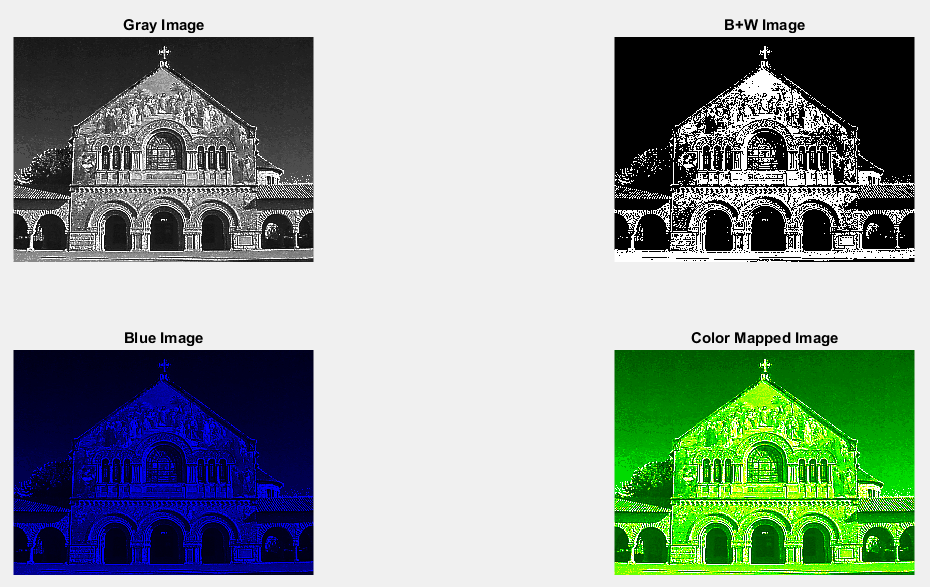
I1 = floor(I/128);

subplot(2,3,6),imshow(mat2gray(I1)),title('1 bit');

**Exercise 3**

This problem was to understand indexed color.

1. By following the same procedure to display image from .dat file the image was shown.
2. As it is known that indexed images consist of 3 color planes so omitting the two will show the remaining colored image. For this purpose, a new plane was created by keeping the blue channel than the image was assigned to that and the resulted image was as shades of blue rather than black/white.
3. Instead of using the Red Green or Blue channel a new color table was to make such that for pixel value 0 to 100 there will be black to green color for 101 to 200 it will become yellow from green and finally from 201 to 255 it will turn into white from yellow. For doing this while checking if there appeared values less than 101 then the red and blue channels were assigned with 0 and the green channel was assigned with values 0 to 1 for 101 to 200 the green channel was assigned 1 and the red channel was assigned from 0 to 1 and finally from 201 to 255 red and green channels were fixed and values were assigned to blue channel. Where there was black color those turned to green and then systematically there was yellow and white color.



The given image is a single channel gray image then binarizing it a pure black and white image is found. Making the image of 3 color plane and omitting others than blue a blue image is found and finally mapping according to problem statement the new color mapped image is found.

**Code:**

% importing .dat file

I = fopen('standford.dat');

% reading .dat file to matlab readable format with defined size

I = fread(I,[580,435]);

% rotating the image

I = imrotate(I,270);

% converting the matrix to gray level

Id = double(I);

Im = mat2gray(Id);

figure;

subplot(2,2,1),imshow(Im),title('Gray Image');

% converting gray image to BW image

Im = imbinarize(Im);

% displaying the BW image

subplot(2,2,2),imshow(Im),title('B+W Image');

% converting gray image to shade of blue

ImBlue(:,:,3) = mat2gray(I);

% displaying the image

subplot(2,2,3),imshow(ImBlue),title('Blue Image');

% applying new color table to the image

ImNew = double(zeros(435,580,3));

for i = 1:435

for j = 1:580

if I(i,j) <= 100

ImNew(i,j,1) = 0;

ImNew(i,j,2) = (Id(i,j)/100);

ImNew(i,j,3) = 0;

elseif I(i,j) <= 200

ImNew(i,j,1) = ((Id(i,j)-100)/100);

ImNew(i,j,2) = 1;

ImNew(i,j,3) = 0;

else

ImNew(i,j,1) = 1;

ImNew(i,j,2) = 1;

ImNew(i,j,3) = ((Id(i,j)-200)/55);

end

end

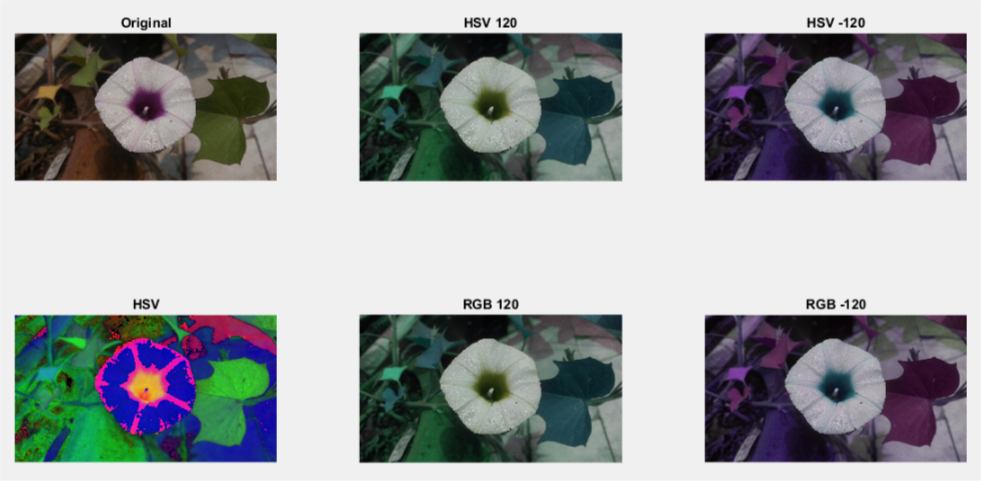
end

% displaying new Image

subplot(2,2,4),imshow(ImNew),title('Color Mapped Image');

**Exercise 4**

This problem was to work with color transformation. First was told to take an image then convert that to HSV using rgb2hsv(). Then the hue of that hsv image was to rotate 120 and neg(120) degree. As the hue component is on the first plane so that was rearranged keeping the other two same. Finally experimentation was done to achieve the same effect of arranging the rgb components.



It becomes clear that the rotating effect of hue can be gained by changing the rgb components values. Though the effect is not gained perfectly but the differences are very low so that that can be ignored.

**Code:**

% reading the rgb image

I = imread('exercise4.jpg');

% displayiing the rgb image

subplot(2,3,1),imshow(I),title('Original');

% converting rgb to hsv

HSV = rgb2hsv(I);

% displaying the hsv image

subplot(2,3,4),imshow(HSV),title('HSV');

% rotation of hue

Hue = HSV(:,:,1)\*360;

HSV(:,:,1) = mod(Hue+120,360)/360;

I120 = hsv2rgb(HSV);

subplot(2,3,2),imshow(I120),title('HSV 120');

HSV(:,:,1) = mod(Hue-120,360)/360;

In120 = hsv2rgb(HSV);

subplot(2,3,3),imshow(In120),title('HSV -120');

% gaining the rotation effect of hue from rgb components

[h, w, d] = size(I);

RGB120 = uint8(zeros(h, w, d));

RGB120(:, :, 1) = I(:, :, 3);

RGB120(:, :, 2) = I(:, :, 1);

RGB120(:, :, 3) = I(:, :, 2);

subplot(2,3,5),imshow(RGB120),title('RGB 120');

RGBn120 = uint8(zeros(h, w, d));

RGBn120(:, :, 1) = I(:, :, 2);

RGBn120(:, :, 2) = I(:, :, 3);

RGBn120(:, :, 3) = I(:, :, 1);

subplot(2,3,6),imshow(RGBn120),title('RGB -120');