DS863 Machine Perception

Assignment 1

ARCHANA R (MT2016021)

CHAVALA SRIHARIPRIYA (MT2016040)

February 1, 2017

Question 1

Choose an RGB image. Plot R, G, and B separately

We read a color image(RGB) and use the split function which divides a multi-channel array into several single-channel arrays. So in this way an RGB image gets divided into separate single Red, Blue, and Green channels.

We plot these images separately. We observe from the below plot *Figure 1* that the Red colored pixels are being separated from the image in the Red channel and similarly for the Green and Blue channels respectively.

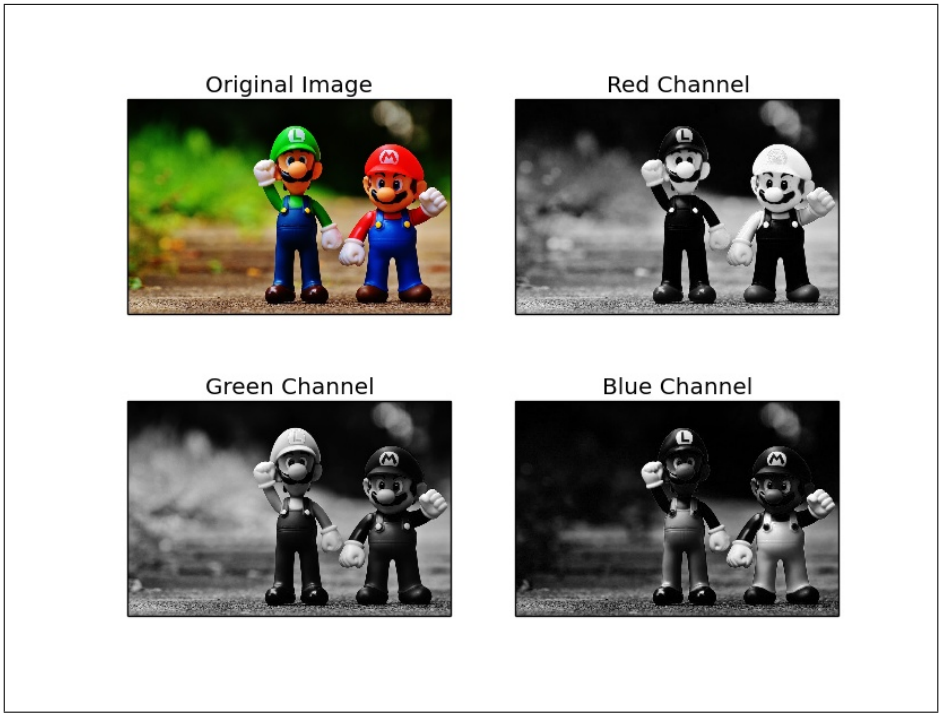

Figure 1: RGB channels

Question 2

Convert an image into HSL and HSV. Write the expressions for computing H, S and V/L

We have taken an RGB image and used pre-defined cvtColor method to convert it to HSV. Then we used split method to split the HSV image into 3 channels(Hue,Saturation and Value). Similarly we used the same methods with different parameter to convert the RGB image to HSL. The expressions used for computing H,S,V and L are as below:

.

R' = R/255,G' = G/255,B' = B/255

Cmax = max(R', G', B'),Cmin = min(R', G', B')

$\Delta = Cmax - Cmin$

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , C_{max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

L=(max(R,G,B)+min(R,G,B))/2

V=max(R,G,B)

The observed images are as shown in Figure 2(H,S and V) and Figure 3(H,S and L).The Hue, Saturation, Value and Luminosity channels are gray scale images as they hold pixel values between 0 to 255.

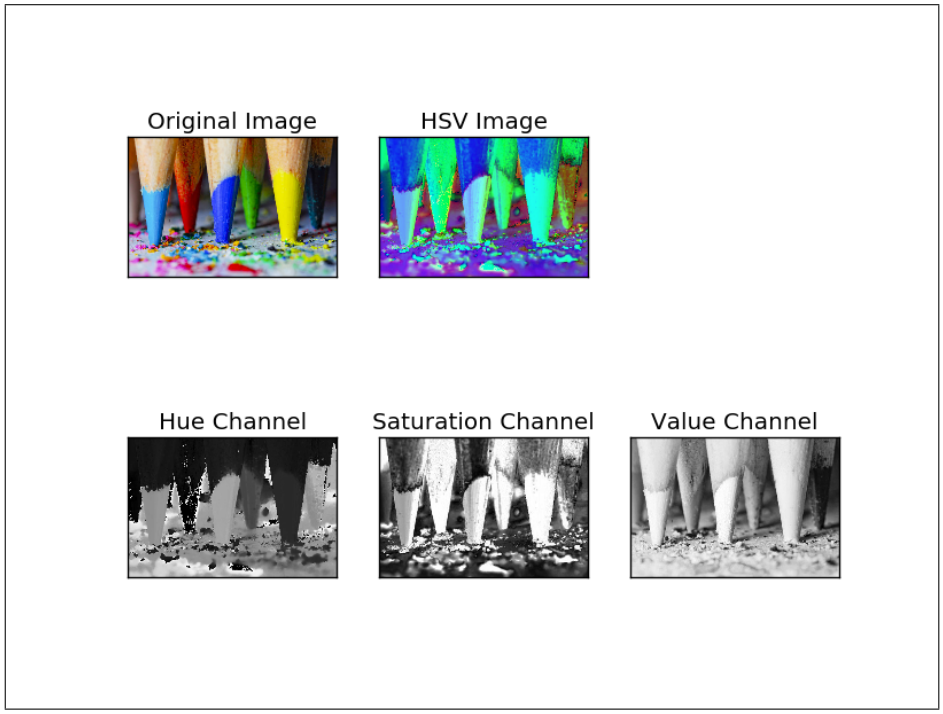

Figure 2: RGB to HSV

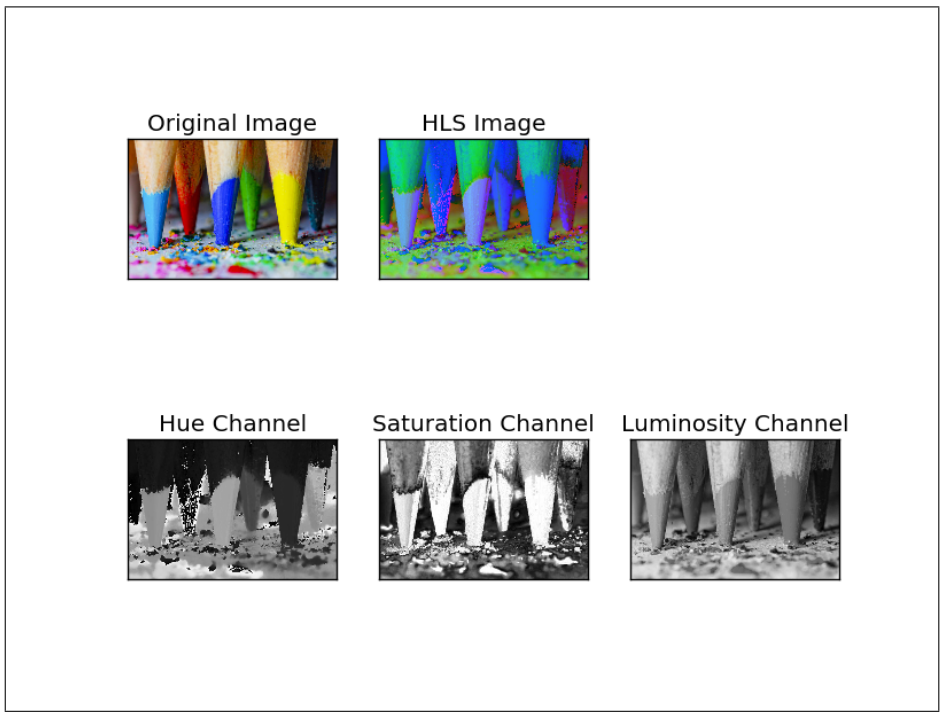

Figure 3: RGB to HLS

Question 3

Convert an image into L*a*b* and plot

We read a color image (RGB) and use a OpenCV's method (*cvtColor*) which is used to convert an image from one color space to another. So in this case we convert from RGB to L*a*b* space, where L* is the lightness component and a* and b* for the color-opponent dimensions based on non linearly compressed (e.g. CIE XYZ) coordinates. The formulae used to convert an image to L*a*b* are as below:

$$
\begin{aligned}
L^\star &= 116 f\left(\frac{Y}{Y_\mathrm{n}}\right) - 16 \\
a^\star &= 500\left(f\left(\frac{X}{X_\mathrm{n}}\right) - f\left(\frac{Y}{Y_\mathrm{n}}\right)\right) \\
b^\star &= 200\left(f\left(\frac{Y}{Y_\mathrm{n}}\right) - f\left(\frac{Z}{Z_\mathrm{n}}\right)\right) \\
f(t) &= \begin{cases} \sqrt[3]{t} & \text{if } t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29} & \text{otherwise} \end{cases} \\
\delta &= \frac{6}{29} \\
X_\mathrm{n} &= 95.047, \\
Y_\mathrm{n} &= 100.000, \\
Z_\mathrm{n} &= 108.883
\end{aligned}
$$

In the below image (*Figure 4*) we plotted the original image along with the transformed L*a*b* and also plotted these transformed channels separately (single-channels).

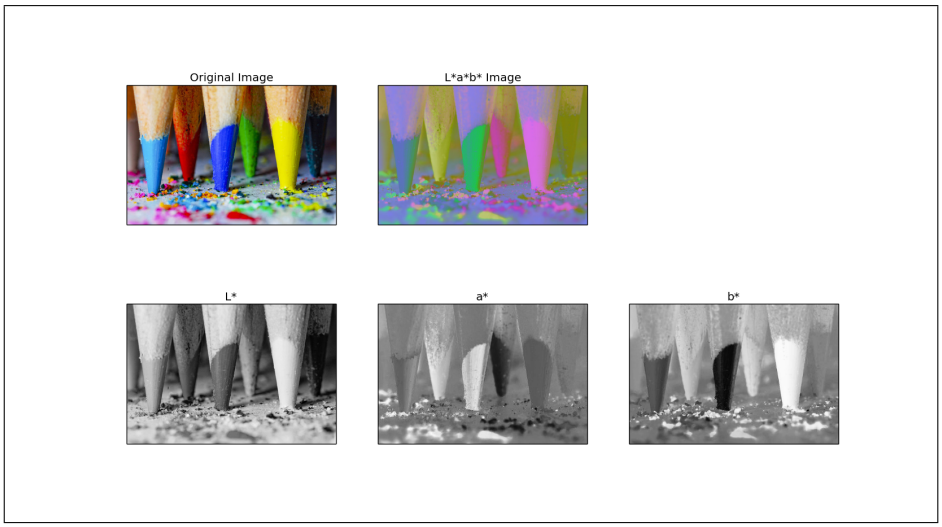

Figure 4: L*a*b*

Question 4

Convert Image 1 into Gray scale using the default OpenCV function. Write the expressions used for the conversion.

Grayscale image is an image where each pixel intensity value is in between 0 to 255. We have taken an RGB image and used predefined cvtColor function with parameter COLOR_BGR2GRAY to convert into Gray scale image. The expression used for conversion is:

gray = 0.30*R+0.59*G+0.11*B

R = red channel value

G = green channel value

B = blue channel value

The results of the image when converted to grayscale using default openCV function is as shown in Figure5.

Figure 5: RGB to Gray

Question 5

Take a grayscale image and illustrate Whitening and Histogram equalization

5a. Whitening

Whitening is one of the image enhancement process where the grayscale image is modified based on mean and standard deviation.
We read a image and convert into grayscale format and apply the below formula to calculate mean and variance of the image.

$$\begin{aligned} \mu &= \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} p_{ij}}{IJ} \\ \sigma^2 &= \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} (p_{ij}-\mu)^2}{IJ}. \end{aligned}$$

Now we apply the below formula to modify the pixels and add them to the original image and then normalize it in the range of (0-255) to obtain the whitened image.

$$x_{ij} = \frac{p_{ij}-\mu}{\sigma}.$$

5b. Histogram equalization

Histogram is graphical representation of the intensity distribution of an image. It quantifies the number of pixels for each intensity value considered. This is done using the below formula.

$$h_k = \sum_{i=1}^{I}\sum_{j=1}^{J} \delta[p_{ij}-k]$$

Histogram equalization is a method that improves the contrast in an image, by stretching out the intensity range. To obtain this we need to calculate the cumulative distribution of the histogram as below

$$c_k = \frac{\sum_{l=1}^{k} h_l}{IJ}$$
$$x_{ij} = K c_{p_{ij}}$$

We observe from the image *Figure 6* that in the histogram equalized image the intensities is uniformly distributed as compared to the whitened image for the same input. We can conclude that Histogram equalization performs better than whitening when the image has many bright pixels since it normalizes the brightness and increases the contrast of the image.

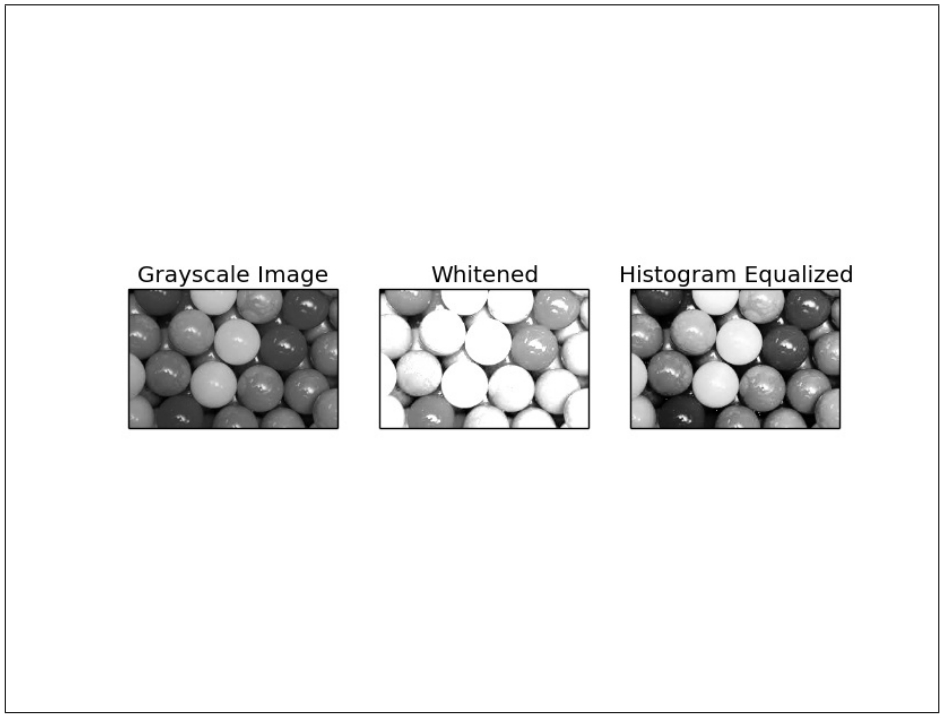

Figure 6: Image Enhancement

Question 6

Take a low illumination noisy image (Image 4), and perform Gaussian smoothing at different scales. What do you observe w.r.t scale variation?

Smoothing is capturing the important patterns of the data and leaving the noise. We have used predefined Gaussian blur method to blur the image and performed smoothing at different scales. Gaussian blur method performs smoothing on image using Gaussian function. It convolute the function over the image and gives the result. The Gaussian function is :

$$G(x) = (1/2\pi\sigma^2)e^{(} - x^2/2\sigma^2)$$

As we increase the scale noise is reduced and also there is a degrade in the quality of image.The results at scales 3 ,7 and 11 are shown in Figure7.The "OpenCV" word in the image contains noise in the original image. As we apply the Gaussian smoothing, the letters are clear and the noise got reduced but the quality is also reduced.

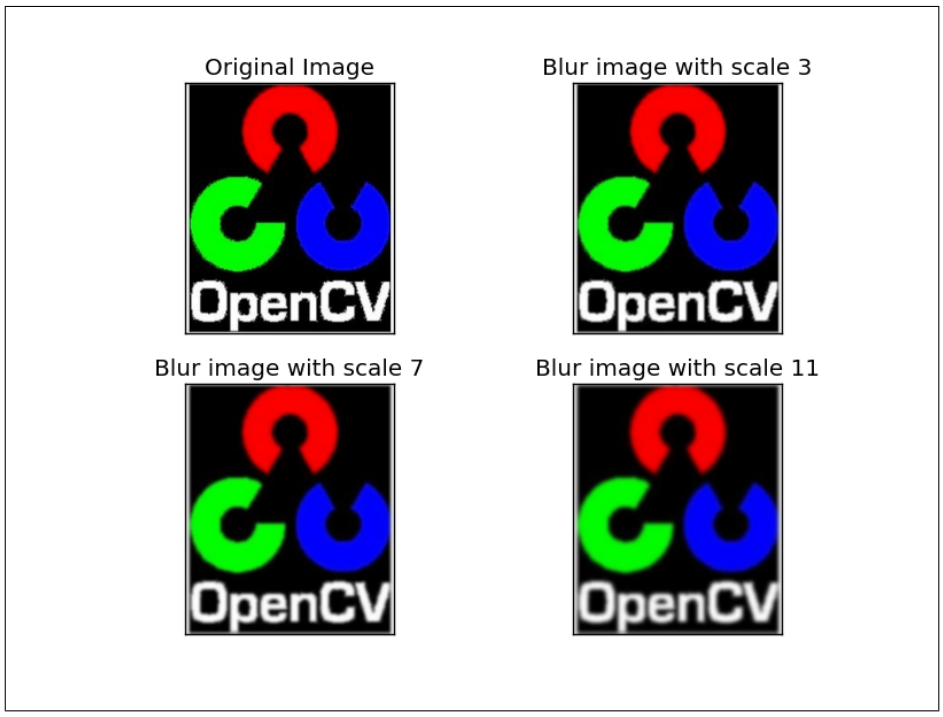

Figure 7: Gaussian blur results at different scales

Question 7

Take an image and add salt-and-pepper noise. Then perform median filtering to remove this noise.

We read an image and convert it to grayscale. To add salt and pepper noise we generate a random number and check if it lies within a given range(limit). If it is below the limit, we make that pixel as a black/pepper noise. If above the limit, it becomes a white/salt noise. Otherwise the pixel is left as is.

Then we use a inbuilt function medianBlur on this noisy image which takes median of all the pixels under kernel area and central element is replaced with this median value. As the kernel size increases the effect of smoothing is increased. Below is the result of median blur with kernel size of 3 on a noisy image

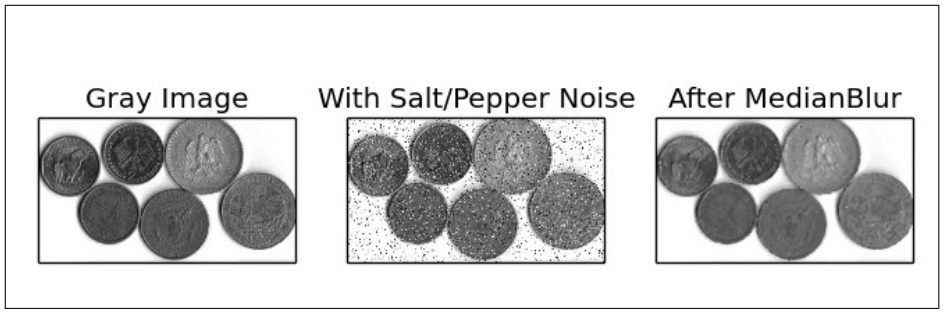

Figure 8: Median Blur on Salt-Pepper noisy image

Question 8

Create binary synthetic images to illustrate the effect of Prewitt (both vertical and horizontal) plus sobel operators (both vertical and horizontal) Clue: check when you have a vertical/horizontal strip of white pixels âĂŞ vary width of the strip from 1 pixel to 5 pixels What do you observe?

We have taken an array and initialized all values to zero to get a black image. Now we have drawn a white strip on that by making corresponding values 255. The convolution on image is performed using filter2D method with the respective masks. The masks are as below:
Prewitt :

$$horizontal = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$vertical = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel :

$$horizontal = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$vertical = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

We have taken a vertical strip on a black image. It is detected when we use horizontal mask of Prewitt and Sobel. To detect a horizontal strip we used vertical masks. After increasing the width of the strip from 1 pixel to 5 pixel wide these operators are detecting top and left edges of the strip but not right and bottom edges. This is because, in masks we are multiplying top, left value with a positive number and bottom and right pixel values with a negative number. So when there is an intensity decrease from left to right or top to bottom then the resulting intensity value after convolution is very low (black pixel) which we can't differentiate on an edge image. The results are shown in Figure 9 and Figure 10.

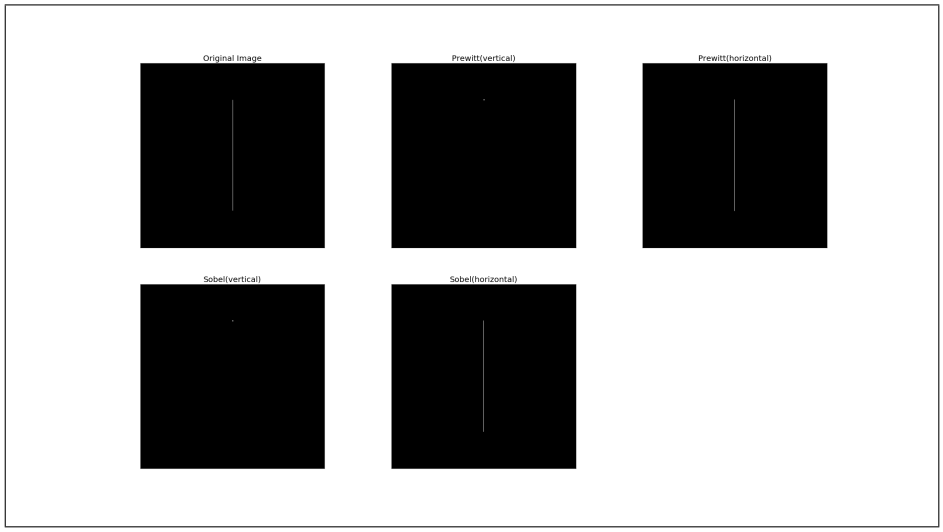

Figure 9: Prewitt and Sobel results on 1 pixel wide strip

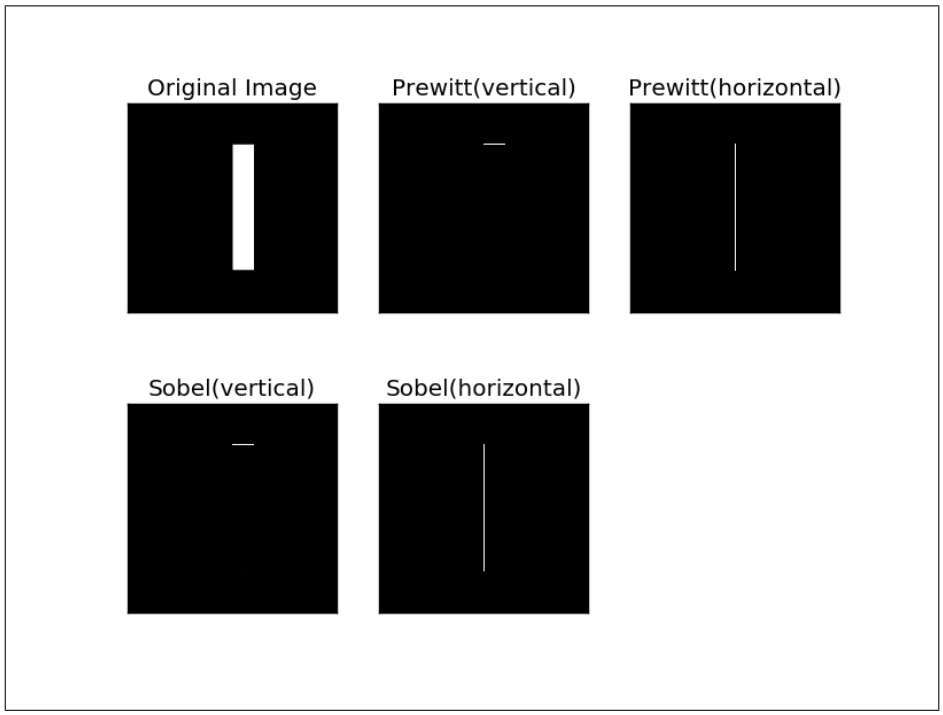

Figure 10: Prewitt and Sobel results on 5 pixel wide strip

Question 9

What filter will you use to detect a strip of 45 degrees

To detect a strip 45° we can make use of Robert's mask.

Robert :

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations.

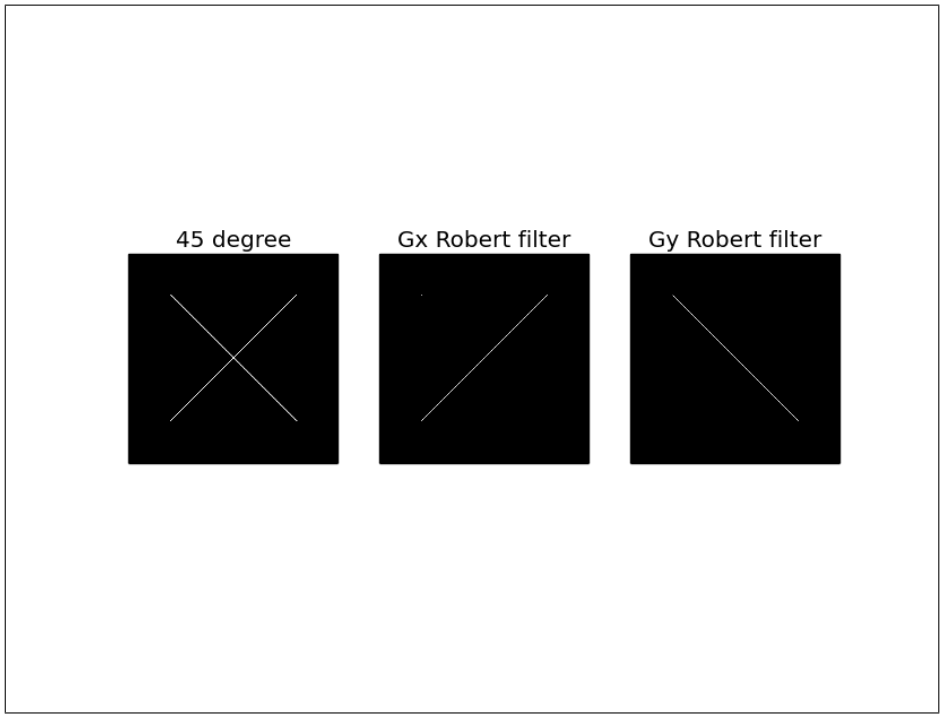

Figure 11: Robert's filter on 45° strips

Question 10

Take an image and observe the effect of Laplacian filtering Can you show edge sharpening using Laplacian edges

Laplacian is a second order derivative used to find the edges of an image. We have taken a RGB image and converted into gray scale image. We have used Gaussian blur on gray scale image as the second order filters are very sensitive to noise. We have used predefined Laplacian method to find the edges of image. The mask used by the method is as below:

$$Laplacianmask = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpening an image is refining the sharp edges of the image. We can refine the edges by increasing their intensity values. So if we add the edge image to the original image, the intensity values at the edges will increased and we will get sharpened edges. Thus

$$\text{sharpened image} = \text{original image} + \text{edge image}$$

The results for edge detection and the sharpening are as shown in Figure12.

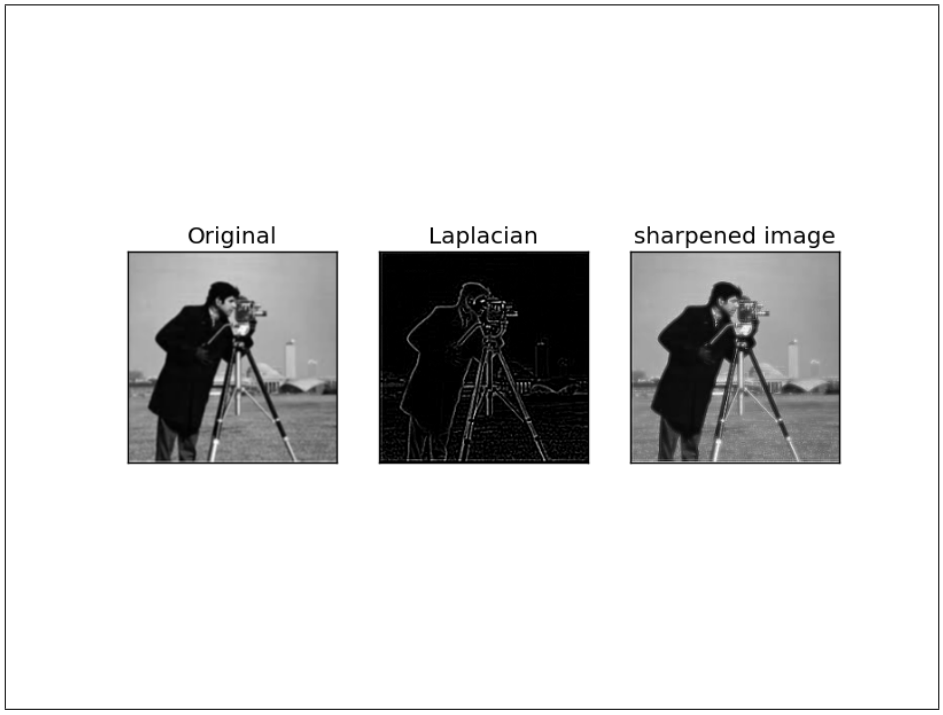

Figure 12: Laplacian edge detection and sharpening

Question 11

Detect Road land markers

Land markers are used to separate the traffic. Yellow lines are used to separate traffic moving in opposite directions, and white lines are used to separate traffic moving in the same direction. So it is very important for an automaton to detect the land markers while driving.

The process of detecting the road markers is as below:

We have taken input images as road containing land markers. We have applied Gaussian Blur on the image to remove the noise so that we can detect the edges properly. To detect the edges we have used the predefined Canny edge detection function in opencv. We have taken minimum threshold as 100 and maximum threshold as 200 to remove the noisy edges in the image. In order to find the land markers we have used HoughlinesP function . This function is used for line detection. The function consists of the following parameters:
rho : Distance resolution of the accumulator in pixels. 1 is used for accurate results
theta : The resolution of the parameter theta in radians. We have used 1 radian
threshold: The minimum number of intersections to detect a line
minLinLength: The minimum number of points that can form a line. Lines with less than this number of points are disregarded
maxLineGap: The maximum gap between two points to be considered in the same line

We have used HoughlinesP function instead of Houghlines because we can specify some parameters like minimum length and maximum line gap of the line. Thus we can omit the lines that are not part of land markers.

The value of the parameters passed to the function HoughLineP will effect the results. So we experimented with different values for parameters threshold,minLinLength and maxLinGap and we have chosen the set of values that gives nearly appropriate results for most of the images. But there will be some outliers where this set of values won't give correct results. The values we have used are threshold = 50, minLinLength = 100 and maxLinGap = 10.

The results of some road images are shown below :

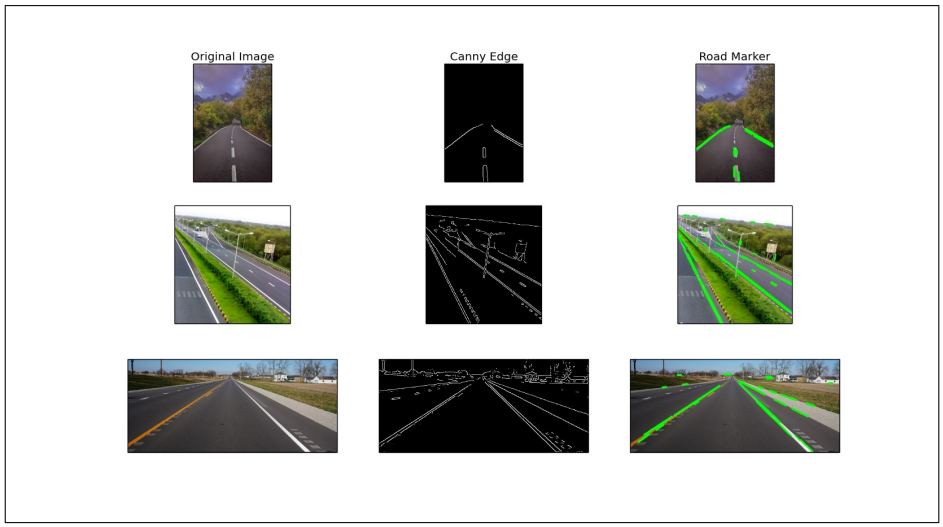

Figure 13: Land markers detection

Question 12

Classify modes: Night; Portrait; Landscape Design features, use NN

There are different modes of images depending on the target as below:
Night : Images which are captured at night
Landscape : Nature images like garden,hills ..etc
Portrait : Images depicting faces

We have used K-nearest neighbour classifier to classify the images into different modes. 3DHistogram is taken as feature for classification. We have used predefined calcHist function. The histogram is extracted from the HSV color space using 8 bins per channel. We have calculated the histograms of training images. These histograms along with the labels are sent to train the KNN-classifier.

The histogram of each test image is calculated.anfind_nearest function is used to predict the label. We have experimented with different values of K and achieved better accuracy at K=5. The training and test data sets are shown in Figure14 and 15.

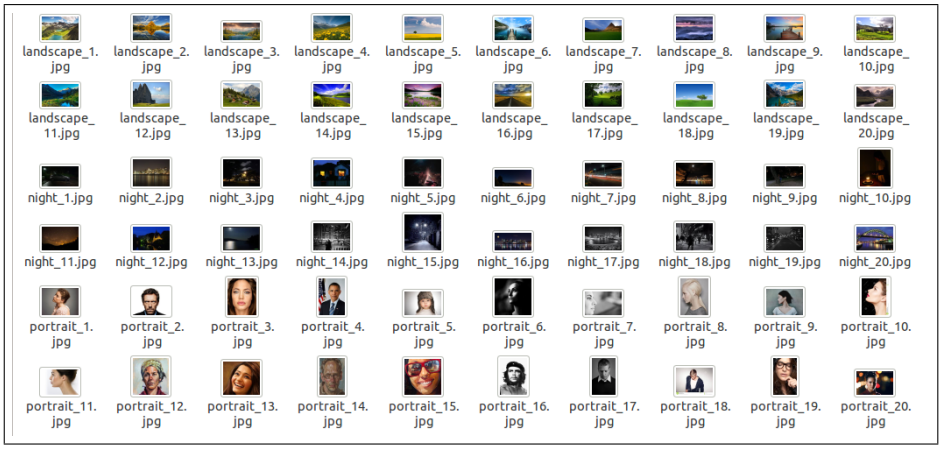

Figure 14: Test data

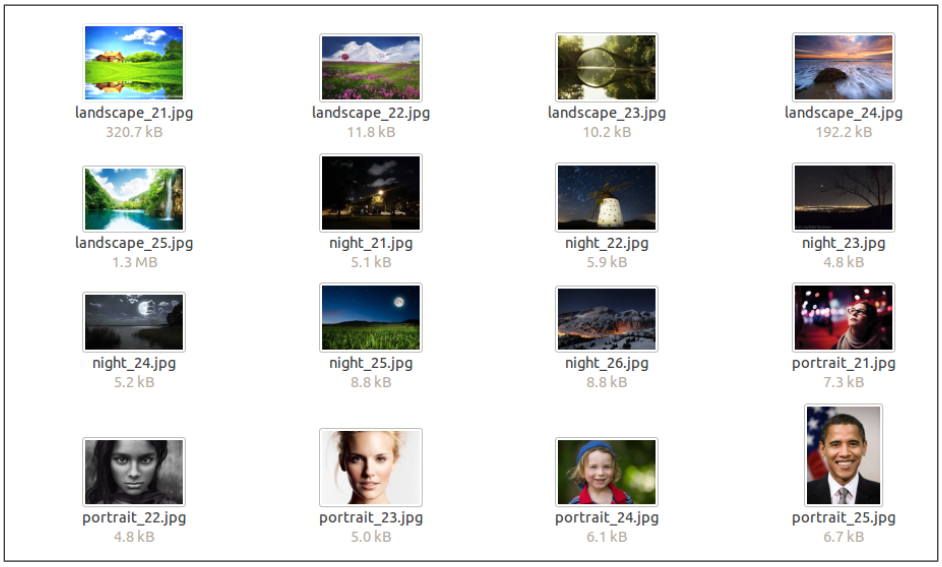

Figure 15: Train data

We have taken a few images which are a combination of 2 or more modes and performed classification and observed that some of them were misclassified. Example : Landscape and night - A nature image captured at night

One of such night images is shown in Figure 16 which was misclassified as landscape.

Figure 16: Misclassified image

There are many pixels whose intensities match to that of the landscape. As we have considered histogram based features, among the top k neighbours majority of them were landscape, so the resulting label is landscape.
The results can be observed in the below image. The accuracy for our data set was observed to be 81.25%

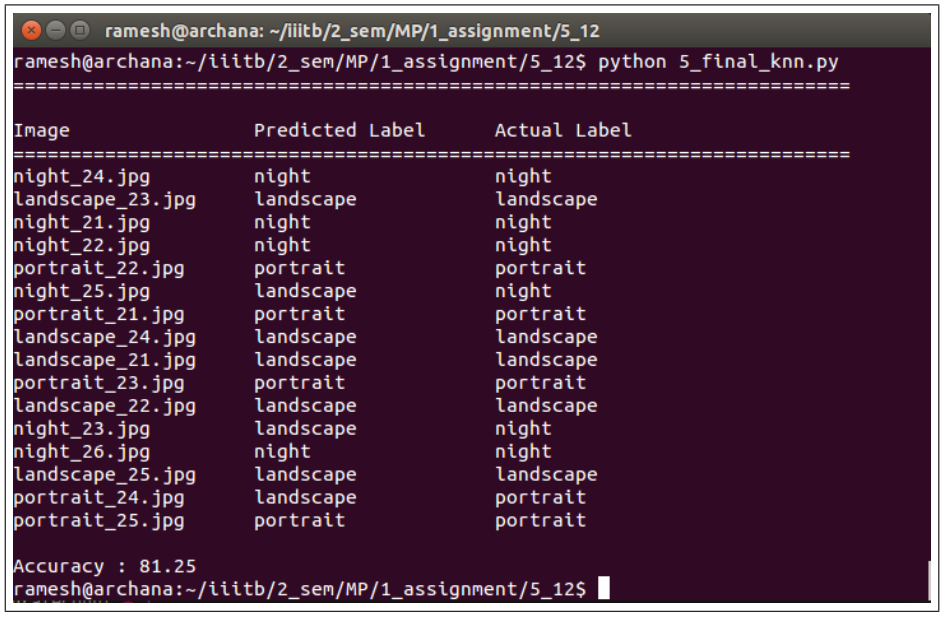

Figure 17: Output