Project # 4 –EECS 490 – DIGITAL IMAGE PROCESSING – FALL '16

Due Date: 10/22/16; Date Submitted: 10/22/16

Color Image Processing

Sai Saradha Kalidaikurichi Lakshmanan[1]

[1]Department of EECS, Case Western Reserve University, Cleveland, Ohio

ABSTRACT:

This report analyzes the fundamentals of color image processing, beginning with the exploration of transforming colors to different spaces, and extending this to segment and detect specific objects in an image. The object under study for this report is orange traffic cones. In the first section of the report, we study two different methods of detecting the cones in the image based on their color. The procedure adapted in the first method is to manually identify samples of the orange color cones in the images and compute the mean and standard deviation of the orange color in these images. We have compared two color spaces, the RGB and YCbCr space and observed that the YCbCr identifies objects better since we remove the illumination component from the image. The second method deals with formulating the conditional probability distribution of the pixels that represent and do not represent a traffic cone. We then test both the models by testing how accurate they are in detecting the cones in a set of test images. Comparison of the two methods indicates that probability distribution more accurately identifies the traffic cones. We also study the significance of histograms by comparing the histograms of the three color channels for different regions in an image and finding how similar or closer they are using the chi-squared distance measure. In the last section, we also analyze two image compression formats – JPEG and PNG, compare their compression techniques and analyze their quality. MATLAB has been used as the programming language.

1. TECHNICAL CONCEPTS

In this section, we review the concepts involved in color image processing.

1.1 Color Spaces

There are different color models that are appropriate for different applications. In this report, we study the HSV and YCbCr color models. When images are taken with variations in light conditions, their intensity profile differs. As a result, if we segment the images directly in the RGB space, the objects are not correctly identified. To handle this, converting the RGB image to other color spaces, such as HSV or YCbCr would help us separate out the illumination component allowing us to work only on the chromaticity of the image. This gives a true distribution of color in the image and can identify the same objects both in light and in dark. Matlab function 'rgb2ycbcr', 'rgb2hsv' does this conversion.

1.2 Average and Standard Deviation metric

One of the methods approached to identify the orange cones in any image is to compute the average and standard deviation of the color in the sample images and check the pixel for this value. Primary step in segmentation is pre-processing. We process the image by filtering the image using Gaussian filter thereby removing noise. We then convert it to the HSV/YCbCr color space and split the image into three individual components, Y, Cb and Cr. Removing the luminance component Y gives us the image with only the color distribution/chromaticity. This method of removing the light intensity works well in identifying objects under different light conditions. For training our model to detect orange traffic cones given any image, samples are collected manually from five color images containing orange cones of different geometric and intensity profiles. The Cb and Cr components are strung out horizontally as a vector and the mean and deviation (Std) are computed. To detect the cones in the new image, all pixels with Cb and Cr value in the range - (Mean± c*Std) where c is the number of deviations from the mean, are identified as cones (shown in fig 1). The code to identify the cones in the image is,

```
for i=1:num_img_ycbcr
    [r c ~]=size(tr_img_ycbcr{1,i});
    tr_img_ycbcr2{1,i}=tr_img_ycbcr{1,i};
    for j=1:r
        for k=1:c
            if((h_lt_ycbcr<=tr_img_ycbcr{1,i}(j,k,2))&&(tr_img_ycbcr{1,i}(j,k,2)<=h_ut_ycbcr)&& (s_lt_ycbcr<=tr_img_ycbcr{1,i}(j,k,3))&&(tr_img_ycbcr{1,i}(j,k,3)<=s_ut_ycbcr))
                tr_img_ycbcr{1,i}(j,k,:)=1;
                tr_img_ycbcr2{1,i}(j,k,:)=tr_img_ycbcr2{1,i}(j,k,:);
            else
                tr_img_ycbcr{1,i}(j,k,:)=0;
                tr_img_ycbcr2{1,i}(j,k,:)=128;
            end
        end
    end
```

Figure 1 Condition to identify a traffic cone using the mean and standard deviation

1.3 Probability distribution and Maximum a Posteriori (MAP) Algorithm

Another method used most commonly in object recognition or identification is the MAP hypothesis. In this method, segmentation is considered as a labeling problem, where the labels are 'is an orange cone' or 'is not an orange cone'. Every pixel can contain only one of these two values, they either are or not a part of the traffic cone and the model is a probability maximization model. From the sample/training images, we first multiply the image with the mask image to get a clear distinction of cone and no cone images, construct two probability distribution functions, one representing the pixels given that they do not represent the cone and the other given that the pixels

represent the cone. The distribution function is obtained by counting the number of pixel combinations of (cb,cr) and dividing them individually by the total number of these values (shown in fig 2). When an unseen image is tested, we consider every pixel and classify that as a cone pixel if the probability that it is a cone color is higher than the probability that it is not.

```
for i = 1:length(h_cone_all_ycbcr_prob)
    Distplot(h_cone_all_ycbcr_prob(i), s_cone_all_ycbcr_prob(i)) = Distplot(h_cone_all_ycbcr_prob(i), s_cone_all_ycbcr_prob(i)) + 1;

end
 prob_dist=Distplot./length(h_cone_all_ycbcr_prob);
%figure, surf(Distplot);

%Distribution for non traffic cones:
for i = 1:length(h_cone_all_ycbcr_prob_nc)
    Distplot_nc(h_cone_all_ycbcr_prob_nc(i), s_cone_all_ycbcr_prob_nc(i)) = Distplot_nc(h_cone_all_ycbcr_prob_nc(i), s_cone_all_ycbcr_prob_nc(i)) + 1;
end
prob_dist_nc=Distplot_nc./length(h_cone_all_ycbcr_prob_nc);
```

Figure 2 Code to identify a cone pixel using probability distribution

1.4 Image Histogram

Histograms play a significant role in providing information about the exposure, saturation of color channels, finding the number of distinct pixel values in an image, etc. Here, we segment the image into three channels, R, G and B, compute the histogram of the three channels individually and then combine them in a single plot. Having them in a single plot allows us to comparatively analyze the properties of the three channels. After we normalize these histograms, chi-squared distance between histograms is computed. This distance measure is an indicator of how similar two regions/images are.

1.5 Image Compression

Image compression can be lossless or lossy and here we study two compression methods, JPEG and PNG. The PNG format is lossless while the JPEG is lossy. We study how information is lost with lossless compression by comparing the histogram of the two images.

2 ANALYSIS

Problem -1

2.1 Segmentation using average and standard deviation metric

As explained in section 1.2, we segmented the image using the average and standard deviation of the cone samples. Fig 3, shows the distribution of the orange color in the sample. It can be noted from the distribution that the orange color is distributed over a very small range of Cb and Cr and these small variations are due to the difference in the color of cones due to numerous factos like illumination or if it is far behind in the image. The original image is Gaussian filtered to remove noise. Then, using the average and deviation, any new image can be segmented. If the Cb and Cr value of a pixel is in the range (mean ± c* std) then we set that pixel to be the cone pixel. As we can see from fig. 4, the orange cones have been accurately identified. We also notice that in some of the images, a small amount of non-cone regions have also been identified. This is because of the similarity in the cb and cr values of the other objects in the scene. However, only a small number of non-cone pixels have been detected and the model is able to identify just the cones neatly.

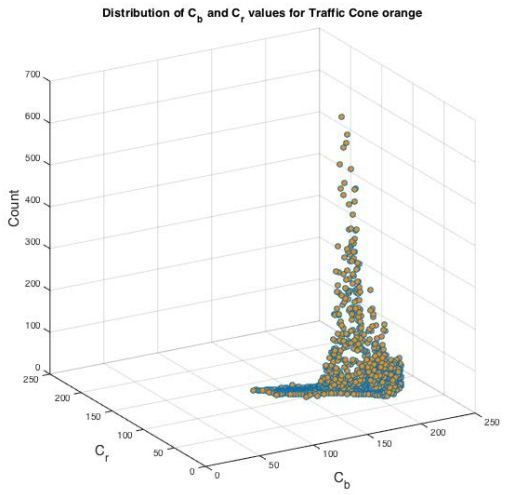
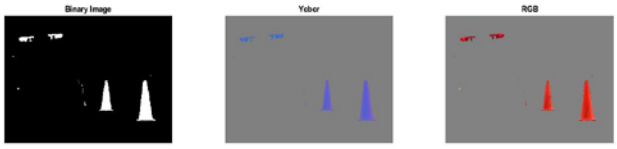
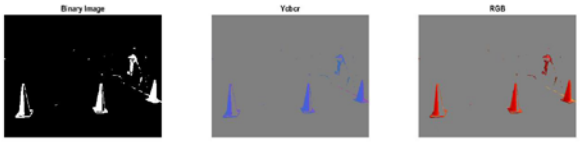


Figure 3: Distribution of Cb and Cr





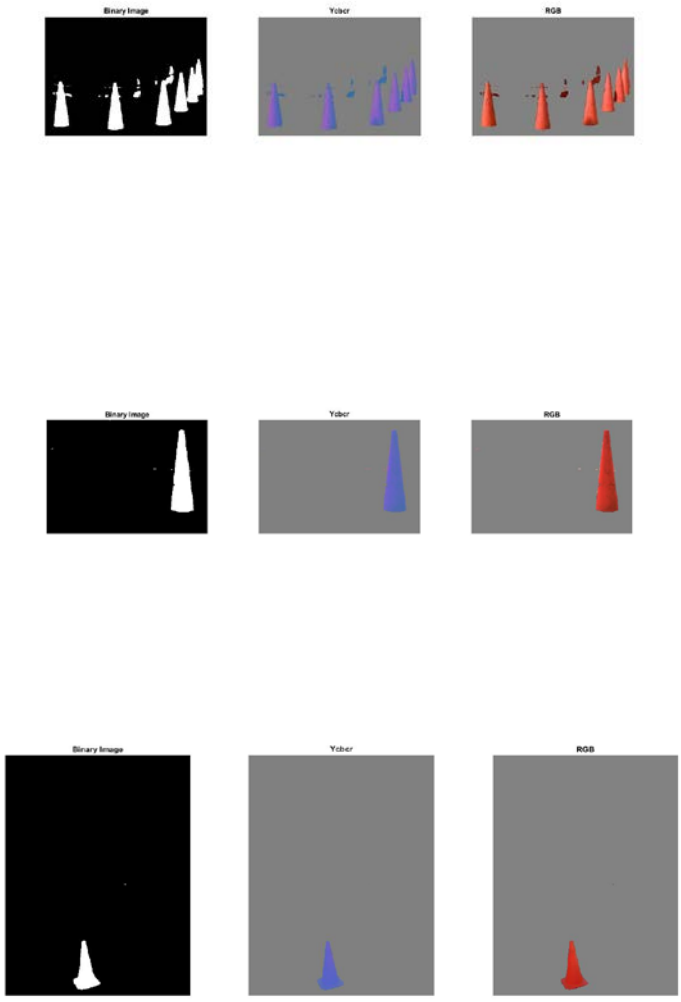


Figure 4: Segmented training images in different color spaces (Binary, YCbCr and RGB)

Fig 5 is the segmentation of the test images in different color spaces. The original test images were not a part of the sample set. The model was able to identify the cones correctly. This is because of the reason that we have removed the light component (Y) from the image, so any variations in orange color intensity because of the light conditions have been handled.

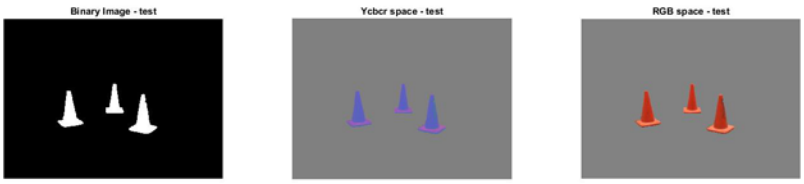


Figure 5: Segemented test images in different color spaces

2.2 MAP Method:

This section deals with the analysis of object detection using the maximum a posteriori algorithm. Multiplying the original image with the mask and the complement of the mask (obtained using imcomplement function in matlab) gives images with just the cones or without the cones.

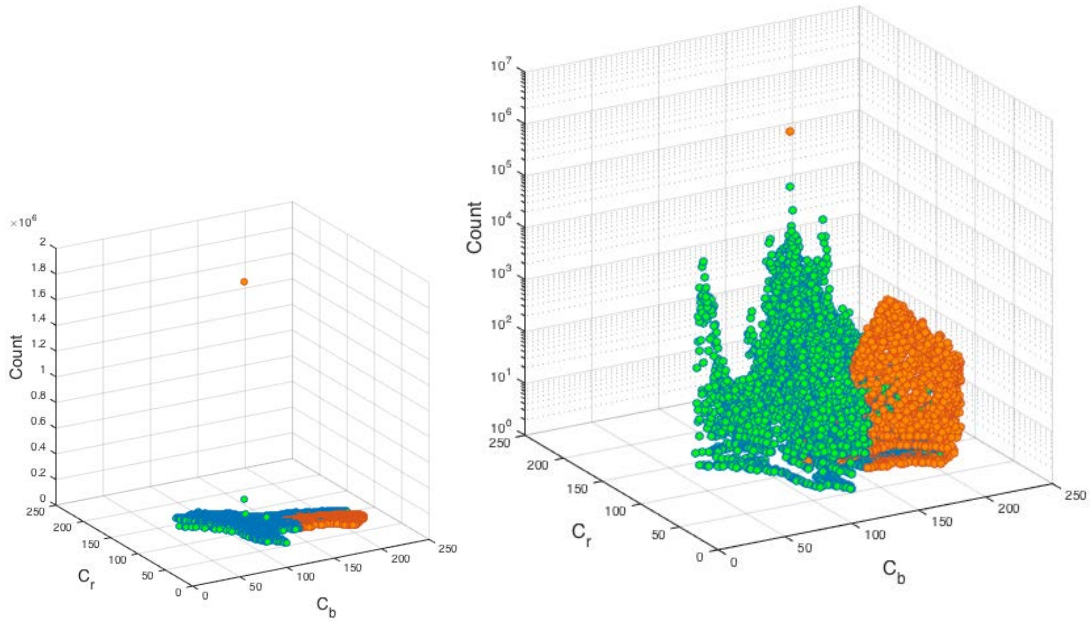


Figure 6: Distribution of (C_b,C_r) values for the traffic cone orange and non-traffic cone regions. The plot on the left is on linear scale while the plot on the right is log-scale in z-axis.

We plot the distribution of C_b and C_r values for traffic cone (color coded as orange filled circles) and non-traffic cone parts (color coded as green filled circles) of the image (fig 6). We find that the distribution for traffic cone and non-traffic cone are fairly well separated where the orange color minimally overlaps with the color belonging to the

non-traffic cone parts. This shows good segemntation between traffic cone and non-traffic cone sections. To further highlight the above point, we re-plot figure in a log-scale for the z-axis. The log scale provides further distinction between traffic cone and non-traffic cone sections. The traffic cone pixel is clearly distinguished with a very high count (~ 10^6) while the rest of the image has a maximum count of only 10^4. The two fold difference further suggests efficient segmentation.

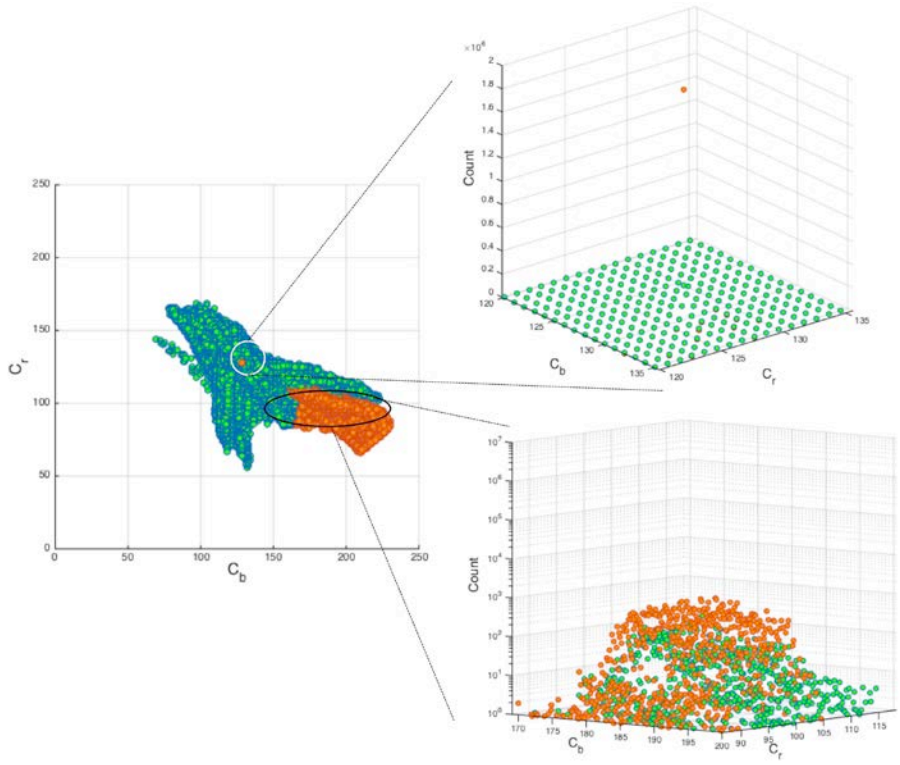
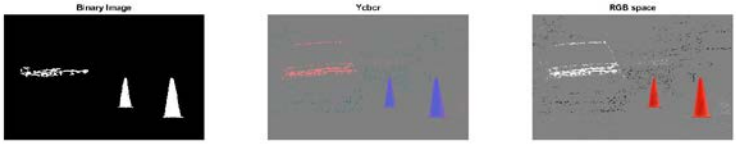
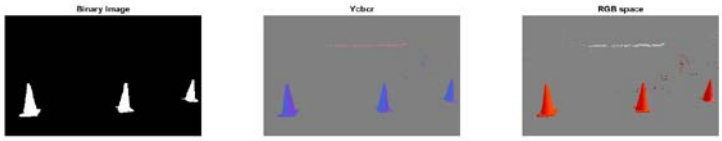


Figure 7: Distribution of (C_b,C_r) values for the traffic cone orange and non-traffic cone regions viewed on 2D surface. The plot on the top-right is a zoom in around the point (128,128) and the plot on the bottom right is a zoom in at the bottm right of the distribution plot.

To further study the above figure, we view the 3D plot as a 2D surface below. We zoom into the neighborhood of the point (128,128) and towards the bottom right of the image where overlap is evident. The orange pixel, belonging to traffic cone at (128,128) has a count of about 2 x 10^6 while pixels belonging to the rest of the image have count utmost 10^4 in that region. At the bottom right of the distribution plot, we see overlap between the traffic cone and non-traffic cone pixels. The maximum count of these overlapped regions is utmost ~3x10^2. Therefore, while there is marginal overlap, the difference in counts differentiates the traffic cone from non-traffic cone clearly.

Using this function, to identify a pixel as a cone pixel, we compare the probability that it is a cone and that it is not a cone. The most likely classification is the one with higher probability. Segmenting the image this way, we are able to identify the cone pixels as shown in fig 8. On the test images too, the cones can be detected nearly accurately (as seen in fig 9). However, we do see reasonable amount of disjoint pixels in the image that have also been detected as cone pixels as they have comparatively higher probability too.


Binary Image
Ycbcr
RGB space


Binary Image
Ycbcr
RGB space

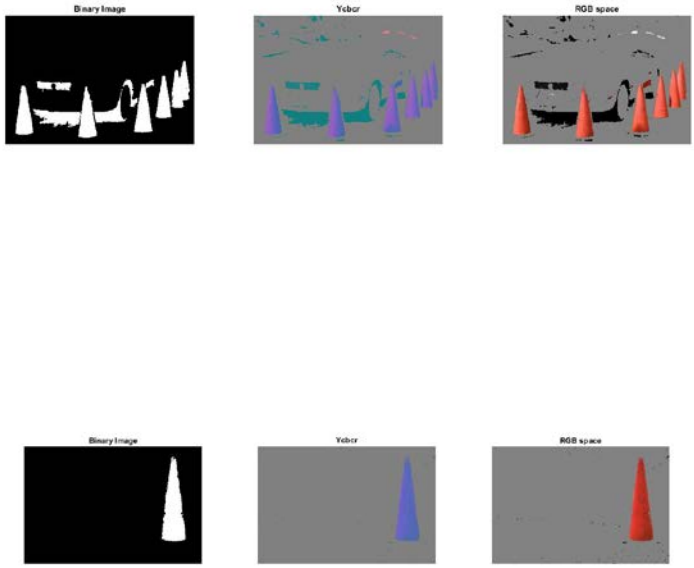
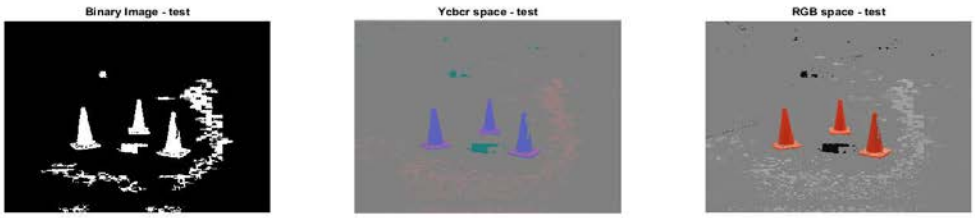


Figure 8: Segmented images using the MAP method. The images shown above are the segmented binary image and the YCbCr and RGB equivalent



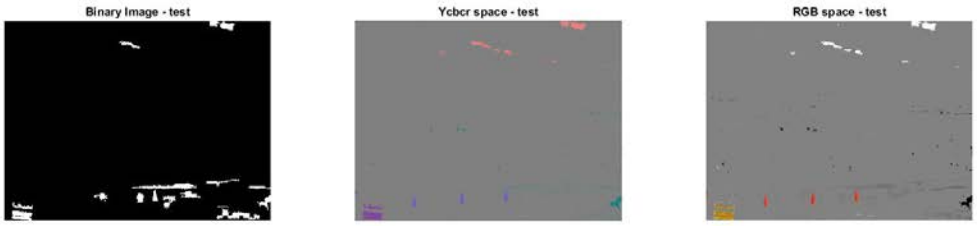


Figure 9: Segmented test images using the MAP method

We can observe that a simple probabilistic method of classifying based on the higher probability produces good results in detecting the cone. We can improve this by using other methods such as maximum likelihood estimation, thresholding etc. To remove the small pixels being identified as cone pixel, we use the function *bwareaopen*, this is a way of identifying a region of connected pixels.

Problem 2

2.3 Histogram and Chi-squared distance

A function was written in Matlab to divide the intensity range of each channel to 32 bins. Since each color can have intensities in the range 0 to 255, splitting that into 32 bins would mean each bin would represent 8 values of intensity. Displaying the histogram of an image or region of an image for three individual color channels is a useful method to do a color level analysis of the image. Fig 10 shows the original color image and the cropped image. Saturation and exposure of a particular color can be identified from this combined histogram. As seen from fig 11, the region selected from the image is mostly green. The histogram shows that red and blue channels are under exposed (left peaks), while green has a right peak indicating that it is over exposed. Thus, a better intuitive understanding of even subtle colors can be obtained from this concatenated histogram. The histograms have been normalized and we can see that they sum up to 1.

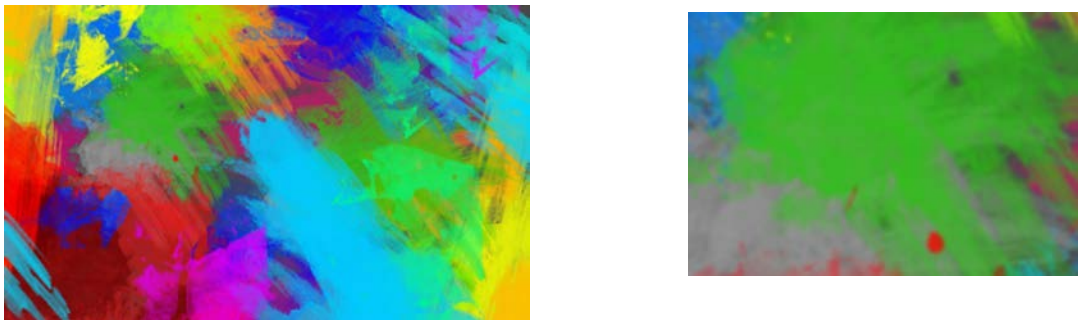


Figure 10. Original and cropped color image

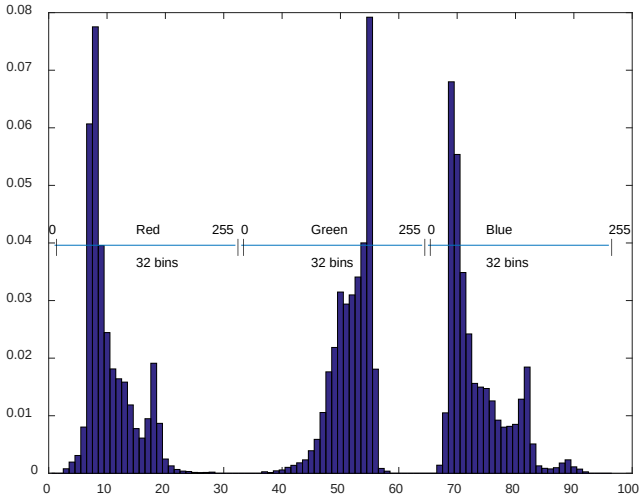
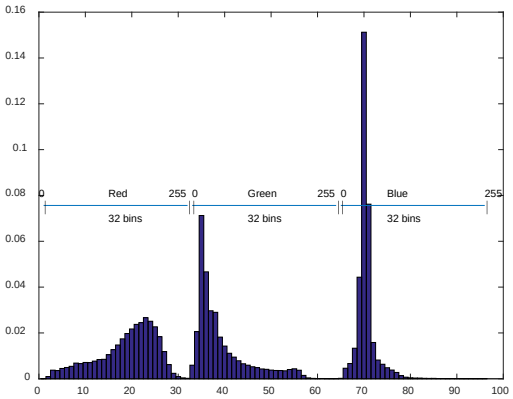
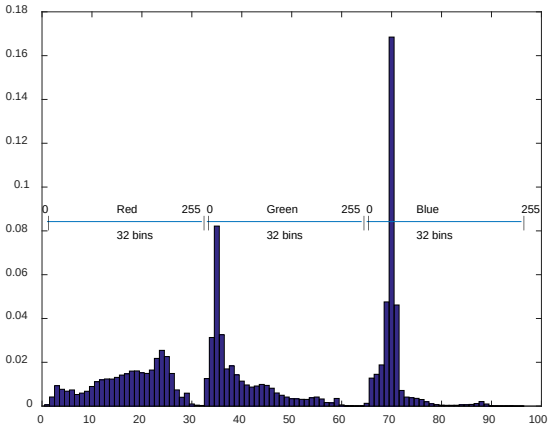
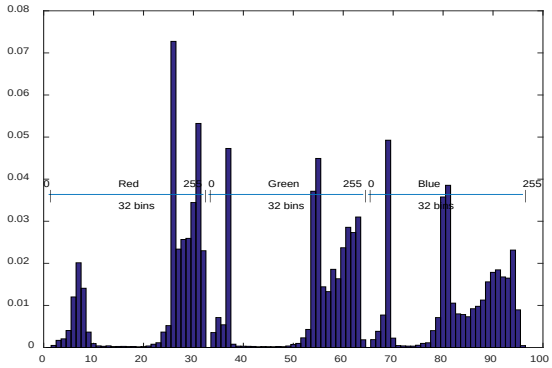


Figure 11. Concatenated histogram of the cropped image

Fig 12 shows the five regions selected from the strawberry and their concatenated histograms. The graphs clearly depict the R, G and B combination of the image. As we can see, within the strawberries, the distribution of R, G and B is different. This is because of the variation in luminance. A peak on the left indicates that most values are near 0 (the absence of that color in the region), while a peak on the right indicates that the image has a large amount of that color component. From the strawberry images it is evident as the peak is near bin 32 (R values for strawberry above 200), while in green and blue the peak is high near bin 0 (the component values are almost 0). From the histogram of the cup, we can see that since white is a dominant color in the image (255,255,255), the peaks are on the right side of the histogram of each color channel.


Red
Green
Blue
255 0
255 0
255
32 bins
32 bins
32 bins


Red
Green
Blue
255 0
255 0
255
32 bins
32 bins
32 bins


Red
Green
Blue
255 0
255 0
255
32 bins
32 bins
32 bins

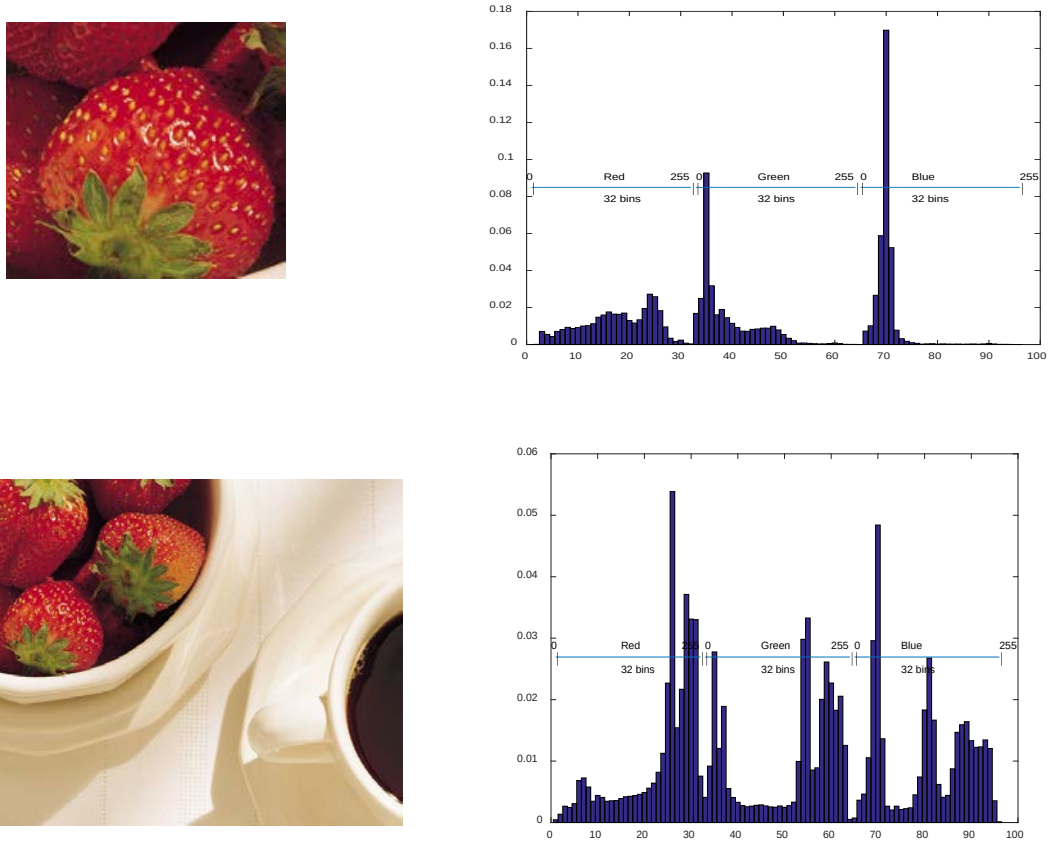


Figure 12. Top to bottom – Images 1 to 5; Five regions from the strawberry and their corresponding histograms

We then implemented a function to compute the squared distance between the histograms. We get a 5x 5 matrix of distance values between every possible pair of histograms. From fig 13, we can see that the matrix is symmetry, that is distance between region 1 and 2 is equal to the distance between regions 2 and 1.From fig 12, images 1 and 2 are very close in that they have the same distribution. Thus the distance is very less (0.0347). The cup is very different from the berries. Thus we can see from the matrix that the distance between the cup and any other image histogram is higher. At the same time, image 3 and 5 have lesser distance (0.1353).

	1	2	3	4	5
1	0	0.0347	0.6832	0.0442	0.3985
2	0.0347	0	0.6748	0.0239	0.3777
3	0.6832	0.6748	0	0.7001	0.1353
4	0.0442	0.0239	0.7001	0	0.4131
5	0.3985	0.3777	0.1353	0.4131	0

Figure 13: Five regions from the strawberry and their corresponding histograms

2.4 Image Compression

We generated the imzoneplate image using the script 'imzoneplate.m' written by Steve Eddins, downloaded from MATLAB Central File Exchange. The generated figure was saved as '.png' file and '.jpeg' file (Fig 14) using 'imwrite'. The resulting PNG image was 219 KB in size and the JPEG image was 73 KB.

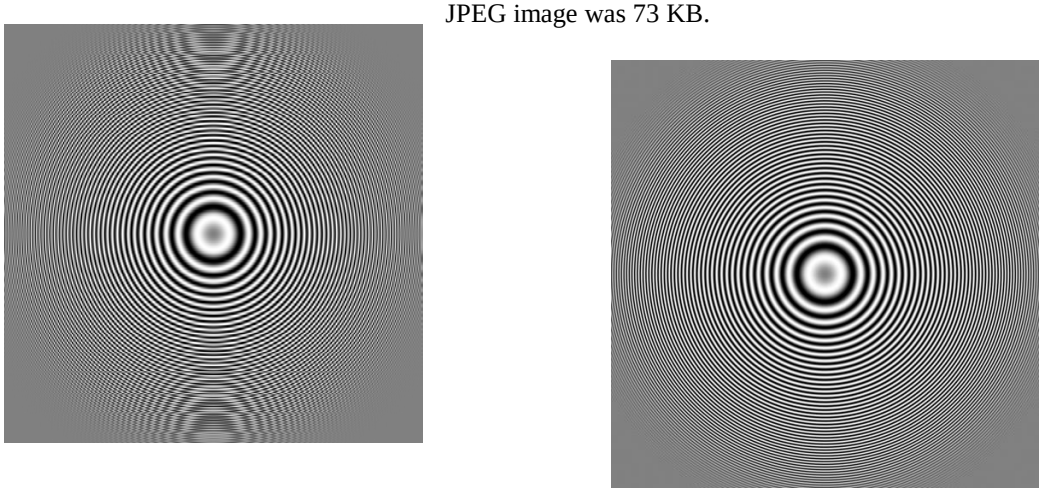


Figure 14: The image 'imzoneplate' in .png (left) and .jpeg (right) file formats

The figure 'imzoneplate' has an intensity profile as shown in Fig.15. The image is radially symmetric with the width of the high intensity region (seen here as peaks) decreasing exponentially as we go away from the center. Therefore, towards the edges, the width of white circle becomes very thin and hence more difficult to resolve.

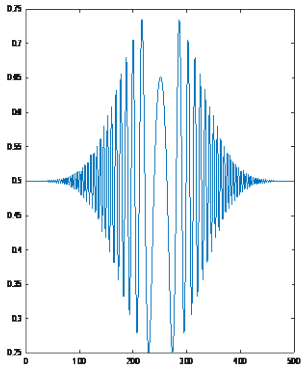


Figure 15: Intensity versus radial distance for an intermediate row of 'imzoneplate'

Hence, resolution of image will be significantly affected by any loss properties of the image file format. From Fig. 14, we note that the PNG file format, visually, has a higher resolution than the equivalent JPEG file format. Specifically, the outer circles (those with lower intensities) appear smeared or discontinuous in the JPEG file format. The difference image (subtraction of JPEG image, from the PNG image, Fig. 14) displayed in Figure 16 highlights this. In Fig. 16, the center spot appears black, indicating that both JPEG and PNG contain the same amount of information at that point. However, as we more radially away from the center, we find that the appearance of more white spots. The white spots represent information contained in PNG that are not contained in the JPEG – that is – information lost in the JPEG compression process.

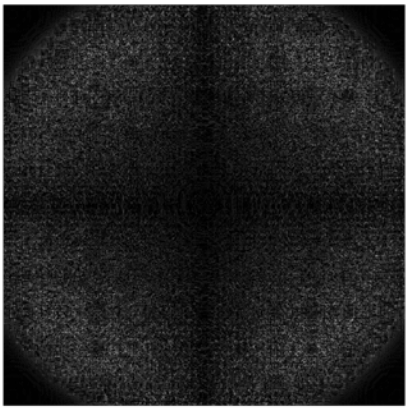


Figure 16. Subtracted image of Figure 2 from Figure 1

The histograms of the distribution of color in the .png version and .jpeg version are plotted in Fig. 17.

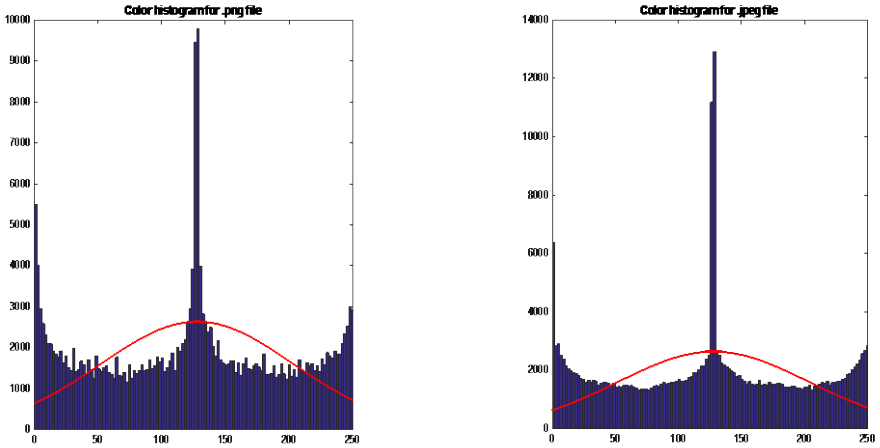


Figure 17. (left) Histogram of color distribution in .png image and **(right)** .jpeg image.

From Fig. 17, we see that, for both '.png' and '.jpeg' compressions, the pixel intensity value of 128 has highest distribution, as expected from Fig. 15. However, the normal distribution (shown here as red) is sharper for the .png image while the .jpeg compression results in a color distribution with a larger variance. That is, .jpeg compression results in an image where the distribution of colors become equally probable, and hence lesser resolution, while the .png compression is able to resolve between the color intensities better resulting in a sharper distribution weighted at the center and hence a sharper image. Figure 18 compares the .jpeg of the zone plate image with quality set to 75 (Fig. 18 left) and with quality parameter set to 25 (Fig. 18 right). The file size when quality parameter was set to 25 was 40 KB, which is lesser than the file size when quality was set to 75. Furthermore, the image with lower quality setting also appeared to be of very poor resolution with the outer circles having visible discontinuities and smear. Figure 19 shows the .jpeg of the zone plate with the quality parameter set to 100. In this case, the file size was 215 KB. Therefore, the file size of the image increased as the quality parameter was increased. Visual inspection shows that the image quality between .jpeg with quality parameter 100 was similar to the .png image.

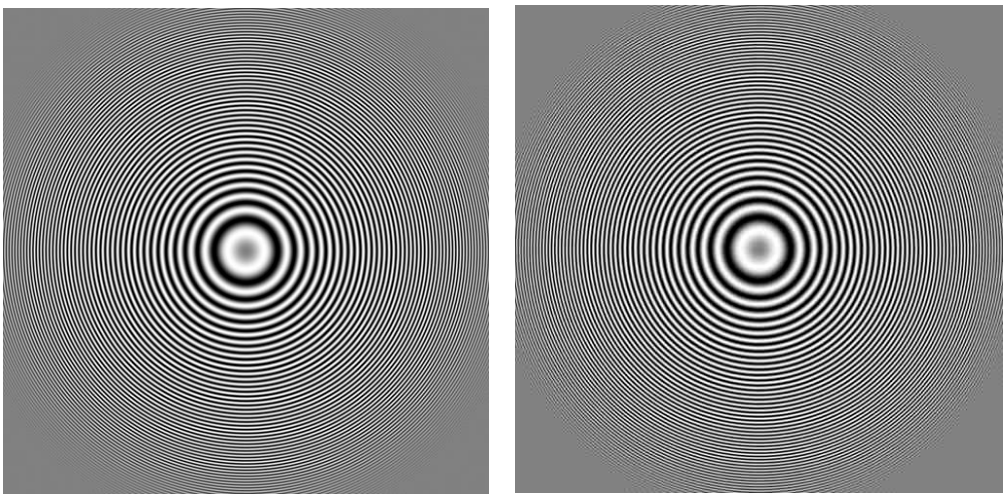


Figure 18: Jpeg image of the zone plate with quality parameter set to 75 **(left)** and 25 **(right)**

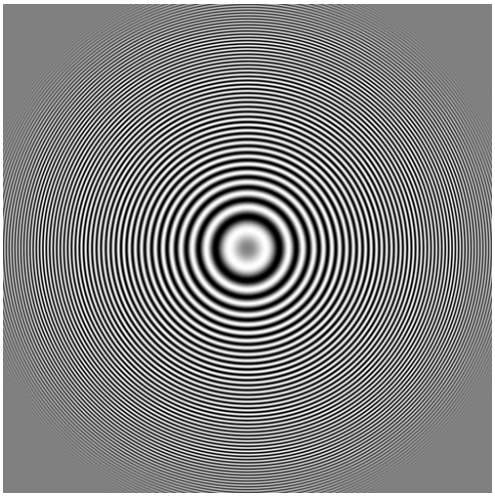


Figure 19. Jpeg image of the zone plate with quality parameter set to 100

Finally, we summarize the above file compression study by plotting and comparing the color distribution histogram for the PNG compression with the JPEG compression with quality settings 100, 75 and 25. The histograms are plotted in Figure 20. As the JPEG quality setting is decreased from 100 to 25, the variance in the color distribution increases and the height of the mean value decreases. This indicates that, with lower quality, the resolution between different intensity values is lost, with every color being 'almost' equally probably. The histogram for the JPEG compression with quality 100 is similar to that of the PNG image. This might indicate that the JPEG compression with quality factor of 100 is lossless compared to the PNG image. However, the file size of the JPEG image with quality factor 100 is 215 KB which is lesser by 4000 bytes than the file size – 219 KB – for the PNG image. Therefore, the JPEG compression with quality factor 100 is **not** lossless compared to the PNG image.

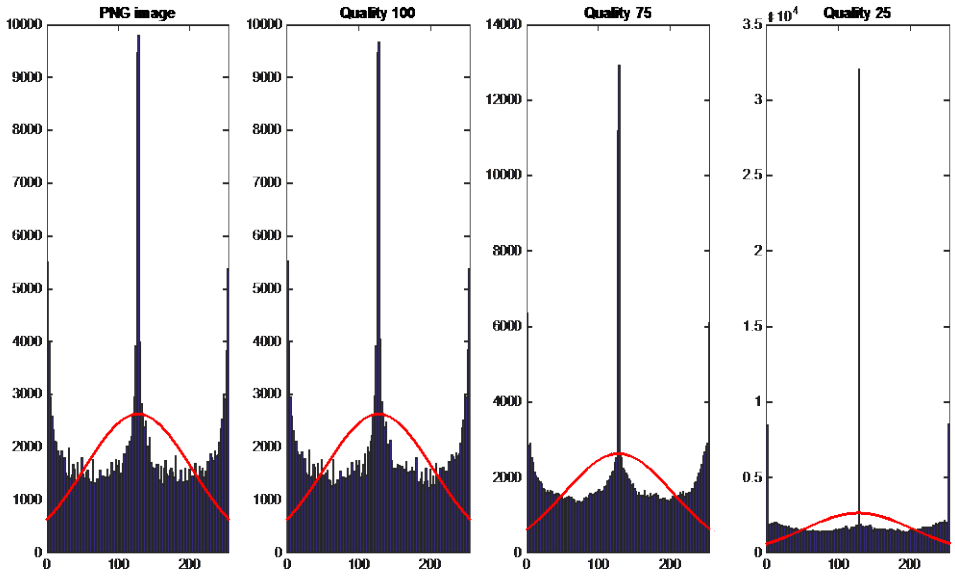


Figure 20. (left – right) Color distribution histogram for .png image, .jpeg image with quality of 100, 75 and 25

The given image was saved as .png and read into MATLAB. The file size of the PNG image was 997 KB. The PNG file was read into MATLAB and then read out as a JPEG file. The file size of the JPEG image was only 64 KB. By comparing the file sizes, the JPEG compression results in a loss by atleast 15 fold. Figure 21 displays the PNG image and JPEG image. Visual inspection and comparison of the two figures show that the PNG image has more information in terms of different shades (light to dark) of purple (of the flower petals). The purple colors in the JPEG image look similar (normalized) across all the petals and information on color shades is lost. This observation is summarized in the color distribution histogram (Fig. 22). The dominant colors in Fig. 21 are green and blue. Figure 22 shows that the PNG image distinguishes between different values of 'green' and 'blue' which is seen as highs and lows in the histogram plot. On the other hand, the JPEG image smears out the color resolution resulting in a range of green and blue colors being equally probably. That is, different shades of green and blue appear in the image with equal probability, and therefore, the information on differences in color shades is lost under JPEG compression. The difference image (Fig. 23) further highlights this difference.

Figure 21. PNG and JPEG (right) image of the plant with flowers

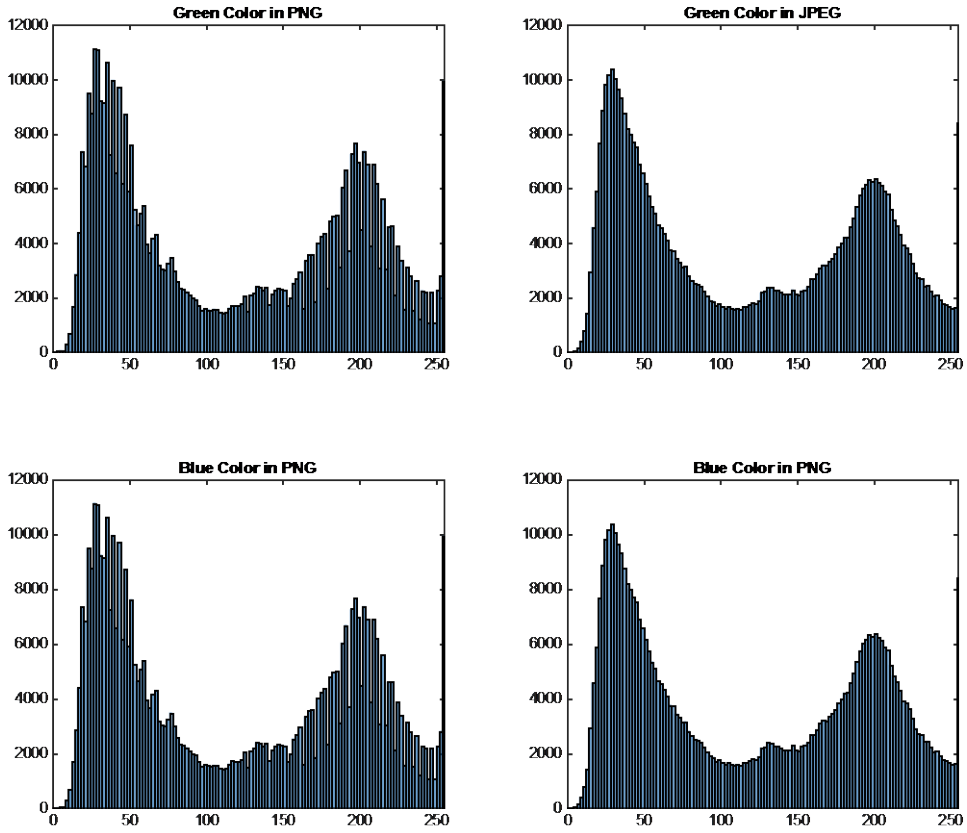


Figure 22. Color distribution histograms of Blue and Green colors in Fig. 8 (PNG image and JPEG image).

Figure 23. Difference image obtained by subtracting JPEG image and from PNG image (Fig. 8)

3. MATLAB ROUTINE

```
%This code demonstrates the techniques of color image processing and
%detection of objects in an image using color; Color histograms and image
%compression methods are also studied
%Code written by Sai Saradha K.L. (MS, Computer Engineering, Fall 2016)


clc;
clear all;

%Collecting samples from the images with traffic cones
cone1_o=imgaussfilt(imread('hw2_cone_training_1.jpg'),1.5);
figure, imshow(cone1_o,[]);
title('Original Cone1');
cone1=rgb2hsv(imcrop(cone1_o));
figure, imshow(cone1,[]);
cone2=rgb2hsv(imcrop(cone1_o));
figure, imshow(cone2,[]);

cone2_o=imgaussfilt(imread('hw2_cone_training_2.jpg'),1.5);
figure, imshow(cone2_o,[]);
title('Original Cone2');
cone3=rgb2hsv(imcrop(cone2_o));
figure, imshow(cone3,[]);
cone4=rgb2hsv(imcrop(cone2_o));
figure, imshow(cone4,[]);
cone5=rgb2hsv(imcrop(cone2_o));
figure, imshow(cone5,[]);

cone3_o=imgaussfilt(imread('hw2_cone_training_3.jpg'),1.5);
figure, imshow(cone3_o,[]);
title('Original Cone3');
cone6=rgb2hsv(imcrop(cone3_o));
figure, imshow(cone6,[]);
cone7=rgb2hsv(imcrop(cone3_o));
figure, imshow(cone7,[]);
cone8=rgb2hsv(imcrop(cone3_o));
figure, imshow(cone8,[]);
cone9=rgb2hsv(imcrop(cone3_o));
figure, imshow(cone9,[]);
cone10=rgb2hsv(imcrop(cone3_o));
figure, imshow(cone10,[]);

cone4_o=imgaussfilt(imread('hw2_cone_training_4.jpg'),1.5);
figure, imshow(cone4_o,[]);
title('Original Cone4');
cone11=rgb2hsv(imcrop(cone4_o));
figure, imshow(cone11,[]);

cone5_o=imgaussfilt(imread('hw2_cone_training_5.jpg'),1.5);
figure, imshow(cone5_o,[]);
title('Original Cone5');
cone12=rgb2hsv(imcrop(cone5_o));
figure, imshow(cone12,[]);
```

```
cone_cell={cone1, cone2, cone3, cone4, cone5, cone6, cone7, cone8, cone9, cone10, cone11, cone12};

%Now we have twelve samples. We need to do the following on these images:
%We need only H and S components, so we separate them out and then
%string them out as a vector
h_cone=zeros;
s_cone=zeros;
h_cone_all=zeros;
s_cone_all=zeros;
h_resh=zeros;
s_resh=zeros;

for i=1:size(cone_cell,2)
   h_resh=cone_cell{1,i}(:,:,1);
   [r,c]=size(h_resh);
   s_resh=cone_cell{1,i}(:,:,2);
   [r1,c1]=size(s_resh);
   h_cone=reshape(cone_cell{1,i}(:,:,1),1,r*c);
   s_cone=reshape(cone_cell{1,i}(:,:,2),1,r1*c1);
   h_cone_all=[h_cone_all h_cone];
   s_cone_all=[s_cone_all s_cone];
end

%Finding the mean and Standard Deviation of the components:
h_mean=mean(h_cone_all);
s_mean=mean(s_cone_all);
h_std=std2(h_cone_all);
s_std=std2(s_cone_all);

%Range for the cone:
h_lt=h_mean-(1*h_std);
h_ut=h_mean+(1*h_std);
s_lt=s_mean-(1*s_std);
s_ut=s_mean+(1*s_std);

%Segmenting the images based on this mean and Std:
%Testing on the training images first:
tr_img_1=rgb2hsv(cone1_o);
tr_img_2=rgb2hsv(cone2_o);
tr_img_3=rgb2hsv(cone3_o);
tr_img_4=rgb2hsv(cone4_o);
tr_img_5=rgb2hsv(cone5_o);

tr_img={tr_img_1,tr_img_2,tr_img_3,tr_img_4,tr_img_5};
num_img=size(tr_img,2);

for i=1:num_img
   [r c ~]=size(tr_img{1,i});
   for j=1:r
      for k=1:c
         if((h_lt<=tr_img{1,i}(j,k,1))&&(tr_img{1,i}(j,k,1)<=h_ut)&&
(s_lt<=tr_img{1,i}(j,k,2))&&(tr_img{1,i}(j,k,2)<=s_ut))
            tr_img{1,i}(j,k,:)=1;
         else
            tr_img{1,i}(j,k,:)=0;
         end
```

```
        end
    end
    figure, imshow(tr_img{1,i});
end

%Now, testing on the test image:
%---- Write code

%Y, Cb, Cr color space:
cone1_ycbcr=rgb2ycbcr(imcrop(cone1_o));
cone2_ycbcr=rgb2ycbcr(imcrop(cone1_o));
cone3_ycbcr=rgb2ycbcr(imcrop(cone2_o));
cone4_ycbcr=rgb2ycbcr(imcrop(cone2_o));
cone5_ycbcr=rgb2ycbcr(imcrop(cone2_o));
cone6_ycbcr=rgb2ycbcr(imcrop(cone3_o));
cone7_ycbcr=rgb2ycbcr(imcrop(cone3_o));
cone8_ycbcr=rgb2ycbcr(imcrop(cone3_o));
cone9_ycbcr=rgb2ycbcr(imcrop(cone3_o));
cone10_ycbcr=rgb2ycbcr(imcrop(cone3_o));
cone11_ycbcr=rgb2ycbcr(imcrop(cone4_o));
cone12_ycbcr=rgb2ycbcr(imcrop(cone5_o));

cone_cell_ycbcr={cone1_ycbcr, cone2_ycbcr, cone3_ycbcr, cone4_ycbcr, cone5_ycbcr, cone6_ycbcr, cone7_ycbcr,
cone8_ycbcr, cone9_ycbcr, cone10_ycbcr, cone11_ycbcr, cone12_ycbcr};

%Now we have twelve samples. We need to do the following on these images:
%We need only H and S components, so we separate them out and then
%string them out as a vector
h_cone_ycbcr=zeros;
s_cone_ycbcr=zeros;
h_cone_all_ycbcr=zeros;
s_cone_all_ycbcr=zeros;
h_resh_ycbcr=zeros;
s_resh_ycbcr=zeros;

for i=1:size(cone_cell_ycbcr,2)
    h_resh_ycbcr=cone_cell_ycbcr{i}(:,:,2);
    [r,c]=size(h_resh_ycbcr);
    s_resh_ycbcr=cone_cell_ycbcr{i}(:,:,3);
    [r1,c1]=size(s_resh_ycbcr);
    h_cone_ycbcr=reshape(cone_cell_ycbcr{i}(:,:,2),1,r*c);
    s_cone_ycbcr=reshape(cone_cell_ycbcr{i}(:,:,3),1,r1*c1);
    h_cone_all_ycbcr=[h_cone_all_ycbcr h_cone_ycbcr];
    s_cone_all_ycbcr=[s_cone_all_ycbcr s_cone_ycbcr];
end

%Finding the mean and Standard Deviation of the components:
h_mean_ycbcr=mean(h_cone_all_ycbcr);
s_mean_ycbcr=mean(s_cone_all_ycbcr);
h_std_ycbcr=std2(h_cone_all_ycbcr);
s_std_ycbcr=std2(s_cone_all_ycbcr);

%Mean and STD Range for the cone:
h_lt_ycbcr=h_mean_ycbcr-(3*h_std_ycbcr);
h_ut_ycbcr=h_mean_ycbcr+(3*h_std_ycbcr);
s_lt_ycbcr=s_mean_ycbcr-(3*s_std_ycbcr);
```

```
s_ut_ycbcr=s_mean_ycbcr+(3*s_std_ycbcr);

%Segmenting the images based on this mean and Std:
%Testing on the training images first:
tr_img_1_ycbcr=rgb2ycbcr(cone1_o);
tr_img_2_ycbcr=rgb2ycbcr(cone2_o);
tr_img_3_ycbcr=rgb2ycbcr(cone3_o);
tr_img_4_ycbcr=rgb2ycbcr(cone4_o);
tr_img_5_ycbcr=rgb2ycbcr(cone5_o);

tr_img_ycbcr={tr_img_1_ycbcr,tr_img_2_ycbcr,tr_img_3_ycbcr,tr_img_4_ycbcr,tr_img_5_ycbcr};
num_img_ycbcr=size(tr_img_ycbcr,2);

tr_img_ycbcr2=cell(1,num_img_ycbcr);
seg_img=cell(1,num_img_ycbcr);
for i=1:num_img_ycbcr
   [r c ~]=size(tr_img_ycbcr{1,i});
   tr_img_ycbcr2{1,i}=tr_img_ycbcr{1,i};
   for j=1:r
      for k=1:c
         if((h_lt_ycbcr<=tr_img_ycbcr{1,i}(j,k,2))&&(tr_img_ycbcr{1,i}(j,k,2)<=h_ut_ycbcr)&&
(s_lt_ycbcr<=tr_img_ycbcr{1,i}(j,k,3))&&(tr_img_ycbcr{1,i}(j,k,3)<=s_ut_ycbcr))
            tr_img_ycbcr{1,i}(j,k,:)=1;
            tr_img_ycbcr2{1,i}(j,k,:)=tr_img_ycbcr2{1,i}(j,k,:);
         else
            tr_img_ycbcr{1,i}(j,k,:)=0;
            tr_img_ycbcr2{1,i}(j,k,:)=128;
         end
      end
   end
   figure, imshow(double(tr_img_ycbcr{1,i}));
   title('Segmented Binary Image');
   figure, imshow(tr_img_ycbcr2{1,i});
   title('Segmented image in Ycbcr space');
   seg_img{1,i}=ycbcr2rgb(tr_img_ycbcr2{1,i});
   figure, imshow(seg_img{1,i});
   title('Segmented image in RGB space');
end

%Now testing on test images:
test_img_1=imgaussfilt(imread('hw2_cone_testing_1.jpg'),1.5);
test_img_2=imgaussfilt(imread('hw2_cone_testing_2.jpg'),1.5);
test_img_1_ycbcr=rgb2ycbcr(test_img_1);
test_img_2_ycbcr=rgb2ycbcr(test_img_2);
test_img_ycbcr={test_img_1_ycbcr,test_img_2_ycbcr};
num_test_img_ycbcr=size(test_img_ycbcr,2);

test_img_ycbcr2=cell(1,num_test_img_ycbcr);
seg_img2=cell(1,num_test_img_ycbcr);
for i=1:num_test_img_ycbcr
   [r c ~]=size(test_img_ycbcr{1,i});
   test_img_ycbcr2{1,i}=test_img_ycbcr{1,i};
   for j=1:r
      for k=1:c
         if((h_lt_ycbcr<=test_img_ycbcr{1,i}(j,k,2))&&(test_img_ycbcr{1,i}(j,k,2)<=h_ut_ycbcr)&&
(s_lt_ycbcr<=test_img_ycbcr{1,i}(j,k,3))&&(test_img_ycbcr{1,i}(j,k,3)<=s_ut_ycbcr))
```

```
            test_img_ycbcr{1,i}(j,k,:)=1;
            test_img_ycbcr2{1,i}(j,k,:)=test_img_ycbcr2{1,i}(j,k,:);
        else
            test_img_ycbcr{1,i}(j,k,:)=0;
            test_img_ycbcr2{1,i}(j,k,:)=128;
        end
      end
   end
   figure, imshow(double(test_img_ycbcr{1,i}));
   title('Segmented Binary Image - test');
   figure, imshow(test_img_ycbcr2{1,i});
   title('Segmented image in Ycbcr space - test');
   seg_img2{1,i}=ycbcr2rgb(test_img_ycbcr2{1,i});
   figure, imshow(seg_img2{1,i});
   title('Segmented image in RGB space - test');
end

%%
%Question - 2nd half - using probability to detect orange color
%Reading the mask images:
mask_img_1=imread('hw2_cone_training_map_1.png');
mask_img_2=imread('hw2_cone_training_map_2.png');
mask_img_3=imread('hw2_cone_training_map_3.png');
mask_img_4=imread('hw2_cone_training_map_4.png');
mask_img_5=imread('hw2_cone_training_map_5.png');

%Since mask image is in png format and training image is in jpeg format, to
%ensure that they are in the proper range:
%Check if this is required:

%Multiplying the original image with the mask
original_cell={cone1_o,cone2_o,cone3_o, cone4_o, cone5_o};
mask_cell={mask_img_1,mask_img_2,mask_img_3,mask_img_4,mask_img_5};
mask_comp_cell={imcomplement(mask_img_1),imcomplement(mask_img_2),imcomplement(mask_img_3),imco
mplement(mask_img_4),imcomplement(mask_img_5)};
size_orig=size(original_cell,2);
size_mask=size(mask_cell,2);
cone_cell_prob=cell(1,size_orig);
nocone_cell_prob=cell(1,size_mask);
cone_cell_ycbcr_prob=cell(1,size_orig);
nocone_cell_ycbcr_prob=cell(1,size_mask);
for i=1:size_orig
    cone_cell_prob{1,i}=original_cell{1,i}.*repmat((uint8(mask_cell{1,i})),[1,1,3]);
%       figure, imshow(cone_cell_prob{1,i});
%       title('Cone only image in RGB space');
    nocone_cell_prob{1,i}=original_cell{1,i}.*repmat((uint8(mask_comp_cell{1,i})),[1,1,3]);
%       figure, imshow(nocone_cell_prob{1,i});
%       title('No Cone image in RGB space');
    cone_cell_ycbcr_prob{1,i}=rgb2ycbcr(cone_cell_prob{1,i});
%       figure, imshow(cone_cell_ycbcr_prob{1,i});
%       title('Cone only image in YCbCr space');
    nocone_cell_ycbcr_prob{1,i}=rgb2ycbcr(nocone_cell_prob{1,i});
%       figure, imshow(nocone_cell_ycbcr_prob{1,i});
%       title('No Cone image in YCbCr space');
end
```

```
h_cone_all_ycbcr_prob=zeros;
s_cone_all_ycbcr_prob=zeros;
h_cone_all_ycbcr_prob_nc=zeros;
s_cone_all_ycbcr_prob_nc=zeros;

for i=1:size(cone_cell_ycbcr_prob,2)
   h_resh_ycbcr_prob=cone_cell_ycbcr_prob{1,i}(:,:,2);
   [r,c]=size(h_resh_ycbcr_prob);
   s_resh_ycbcr_prob=cone_cell_ycbcr_prob{1,i}(:,:,3);
   [r1,c1]=size(s_resh_ycbcr_prob);
   h_cone_ycbcr_prob=reshape(h_resh_ycbcr_prob,1,r*c);
   s_cone_ycbcr_prob=reshape(s_resh_ycbcr_prob,1,r1*c1);
   h_cone_all_ycbcr_prob=[h_cone_all_ycbcr_prob h_cone_ycbcr_prob];
   s_cone_all_ycbcr_prob=[s_cone_all_ycbcr_prob s_cone_ycbcr_prob];
end

for i=1:size(nocone_cell_ycbcr_prob,2)
   h_resh_ycbcr_prob_nc=nocone_cell_ycbcr_prob{1,i}(:,:,2);
   [r2,c2]=size(h_resh_ycbcr_prob_nc);
   s_resh_ycbcr_prob_nc=nocone_cell_ycbcr_prob{1,i}(:,:,3);
   [r3,c3]=size(s_resh_ycbcr_prob_nc);
   h_cone_ycbcr_prob_nc=reshape(h_resh_ycbcr_prob_nc,1,r2*c2);
   s_cone_ycbcr_prob_nc=reshape(s_resh_ycbcr_prob_nc,1,r3*c3);
   h_cone_all_ycbcr_prob_nc=[h_cone_all_ycbcr_prob_nc h_cone_ycbcr_prob_nc];
   s_cone_all_ycbcr_prob_nc=[s_cone_all_ycbcr_prob_nc s_cone_ycbcr_prob_nc];
end

%Eliminating the 0s in the vector for calculating the distribution:
h_cone_all_ycbcr_prob=h_cone_all_ycbcr_prob(h_cone_all_ycbcr_prob~=0);
s_cone_all_ycbcr_prob=s_cone_all_ycbcr_prob(s_cone_all_ycbcr_prob~=0);
h_cone_all_ycbcr_prob_nc=h_cone_all_ycbcr_prob_nc(h_cone_all_ycbcr_prob_nc~=0);
s_cone_all_ycbcr_prob_nc=s_cone_all_ycbcr_prob_nc(s_cone_all_ycbcr_prob_nc~=0);

%Formulating the distributions:
%Distribution for traffic cones:
Distplot= zeros(256);
Distplot_nc=zeros(256);
h_cone_all_ycbcr_prob = round(h_cone_all_ycbcr_prob);
s_cone_all_ycbcr_prob = round(s_cone_all_ycbcr_prob);
h_cone_all_ycbcr_prob_nc=round(h_cone_all_ycbcr_prob_nc);
s_cone_all_ycbcr_prob_nc=round(s_cone_all_ycbcr_prob_nc);
for i = 1:length(h_cone_all_ycbcr_prob)
  Distplot(h_cone_all_ycbcr_prob(i), s_cone_all_ycbcr_prob(i)) = Distplot(h_cone_all_ycbcr_prob(i),
s_cone_all_ycbcr_prob(i)) + 1;

end
 prob_dist=Distplot./length(h_cone_all_ycbcr_prob);
figure, surf(Distplot);

%Distribution for non traffic cones:
for i = 1:length(h_cone_all_ycbcr_prob_nc)
  Distplot_nc(h_cone_all_ycbcr_prob_nc(i), s_cone_all_ycbcr_prob_nc(i)) =
Distplot_nc(h_cone_all_ycbcr_prob_nc(i), s_cone_all_ycbcr_prob_nc(i)) + 1;
end
prob_dist_nc=Distplot_nc./length(h_cone_all_ycbcr_prob_nc);
figure, surf(Distplot_nc);
```

```
%Now segmenting the image based on the probability distribution:
%We now have five training images and two test images in YCbCr space
%tr_img_ycbcr={tr_img_1_ycbcr,tr_img_2_ycbcr,tr_img_3_ycbcr,tr_img_4_ycbcr,tr_img_5_ycbcr};
%test_img_ycbcr={test_img_1_ycbcr,test_img_2_ycbcr};
tr_img_1_ycbcr=rgb2ycbcr(cone1_o);
tr_img_2_ycbcr=rgb2ycbcr(cone2_o);
tr_img_3_ycbcr=rgb2ycbcr(cone3_o);
tr_img_4_ycbcr=rgb2ycbcr(cone4_o);
tr_img_5_ycbcr=rgb2ycbcr(cone5_o);

tr_img_ycbcr={tr_img_1_ycbcr,tr_img_2_ycbcr,tr_img_3_ycbcr,tr_img_4_ycbcr,tr_img_5_ycbcr};
num_img_ycbcr=size(tr_img_ycbcr,2);

tr_img_ycbcr2=cell(1,num_img_ycbcr);
seg_img=cell(1,num_img_ycbcr);

for i=1:num_img_ycbcr
   [r c ~]=size(tr_img_ycbcr{1,i});
   tr_img_ycbcr2{1,i}=tr_img_ycbcr{1,i};
   for j=1:r
      for k=1:c
         cb_val=tr_img_ycbcr{1,i}(j,k,2);
         cr_val=tr_img_ycbcr{1,i}(j,k,3);
         if(prob_dist(cb_val,cr_val)>prob_dist_nc(cb_val,cr_val))
            tr_img_ycbcr{1,i}(j,k,:)=1;
            tr_img_ycbcr2{1,i}(j,k,:)=tr_img_ycbcr2{1,i}(j,k,:);
         else
            tr_img_ycbcr{1,i}(j,k,:)=0;
            tr_img_ycbcr2{1,i}(j,k,:)=128;
         end
      end
   end
   tr_img_ycbcr{1,i}=bwareaopen(tr_img_ycbcr{1,i},5000);
   figure, imshow(double(tr_img_ycbcr{1,i}));
   title('Segmented Binary Image');
   figure, imshow(tr_img_ycbcr2{1,i});
   title('Segmented image in Ycbcr space');
   seg_img{1,i}=ycbcr2rgb(tr_img_ycbcr2{1,i});
   figure, imshow(seg_img{1,i});
   title('Segmented image in RGB space');
end

%Now testing on test images:
test_img_1=imgaussfilt(imread('hw2_cone_testing_1.jpg'),1.5);
test_img_2=imgaussfilt(imread('hw2_cone_testing_2.jpg'),1.5);
test_img_1_ycbcr=rgb2ycbcr(test_img_1);
test_img_2_ycbcr=rgb2ycbcr(test_img_2);
test_img_ycbcr={test_img_1_ycbcr,test_img_2_ycbcr};
num_test_img_ycbcr=size(test_img_ycbcr,2);

test_img_ycbcr2=cell(1,num_test_img_ycbcr);
seg_img2=cell(1,num_test_img_ycbcr);
for i=1:num_test_img_ycbcr
   [r c ~]=size(test_img_ycbcr{1,i});
   test_img_ycbcr2{1,i}=test_img_ycbcr{1,i};
```

```
    for j=1:r
        for k=1:c
            cb_val=test_img_ycbcr{1,i}(j,k,2);
            cr_val=test_img_ycbcr{1,i}(j,k,3);
            if(prob_dist(cb_val,cr_val)>=prob_dist_nc(cb_val,cr_val))
                test_img_ycbcr{1,i}(j,k,:)=1;
                test_img_ycbcr2{1,i}(j,k,:)=test_img_ycbcr2{1,i}(j,k,:);
            else
                test_img_ycbcr{1,i}(j,k,:)=0;
                test_img_ycbcr2{1,i}(j,k,:)=128;
            end
        end
    end
    test_img_ycbcr{1,i}=bwareaopen(test_img_ycbcr{1,i},1000);
    figure, imshow(double(test_img_ycbcr{1,i}));
    title('Segmented Binary Image - test');
    figure, imshow(test_img_ycbcr2{1,i});
    title('Segmented image in Ycbcr space - test');
    seg_img2{1,i}=ycbcr2rgb(test_img_ycbcr2{1,i});
    figure, imshow(seg_img2{1,i});
    title('Segmented image in RGB space - test');
end

%Routine to plot distribution of orange color from traffic cone images%
hsize = size(h_cone_all_ycbcr);
ssize = size(s_cone_all_ycbcr);
count_hs = zeros(256,256);
for i=1:hsize(1,2)
    if h_cone_all_ycbcr(1,i)~=0 && s_cone_all_ycbcr(1,i)~=0

count_hs(h_cone_all_ycbcr(1,i),s_cone_all_ycbcr(1,i))=count_hs(h_cone_all_ycbcr(1,i),s_cone_all_ycbcr(1,i))+1;
    end
end

l=1;
for i=1:256
    for j=1:256
        if count_hs(i,j)~=0;
            X(l,1)=i;
            Y(l,1)= j;
            Z(l,1)=count_hs(i,j);
            l=l+1;
        end
     end
end
 figure, m = scatter3(Y,X,Z);
 xlim([0 250]),ylim([0 250]);
 axis('square');view(-30,20);
 title('Distribution of C_{b} and C_{r} values for Traffic Cone orange','FontSize',13);
 xlabel('C_{b}','FontSize',15);ylabel('C_{r}','FontSize',15);zlabel('Count','FontSize',15);
 m.MarkerFaceColor = [255/255 140/255 0/255];

 %Routine to plot PDF of orange for traffic cone and non-traffic cone%
 hcsize = size(h_cone_all_ycbcr_prob);
 scsize = size(s_cone_all_ycbcr_prob);
 hncsize = size(h_cone_all_ycbcr_prob_nc);
```

```
 sncsize = size(s_cone_all_ycbcr_prob_nc);
 count_hsc = zeros(256,256);
 count_hsnc = zeros(256,256);
for i=1:hcsize(1,2)
    if h_cone_all_ycbcr_prob(1,i)~=0 && s_cone_all_ycbcr_prob(1,i)~=0

count_hsc(h_cone_all_ycbcr_prob(1,i),s_cone_all_ycbcr_prob(1,i))=count_hsc(h_cone_all_ycbcr_prob(1,i),s_cone_
all_ycbcr_prob(1,i))+1;
    end
end

for i=1:hncsize(1,2)
    if h_cone_all_ycbcr_prob_nc(1,i)~=0 && s_cone_all_ycbcr_prob_nc(1,i)~=0

count_hsnc(h_cone_all_ycbcr_prob_nc(1,i),s_cone_all_ycbcr_prob_nc(1,i))=count_hsnc(h_cone_all_ycbcr_prob_n
c(1,i),s_cone_all_ycbcr_prob_nc(1,i))+1;
    end
end

X = zeros;
Y=zeros;
Z=zeros;
l=1;
for i=1:256
    for j=1:256
        if count_hsc(i,j)~=0;
            X(l,1)=i;
            Y(l,1)= j;
            Z(l,1)=count_hsc(i,j);
            l=l+1;
        end
     end
end
l=1;
for i=1:256
    for j=1:256
        if count_hsnc(i,j)~=0;
            Xnc(l,1)=i;
            Ync(l,1)= j;
            Znc(l,1)=count_hsnc(i,j);
            l=l+1;
        end
     end
end

 figure, m = scatter3(Ync,Xnc,Znc,'MarkerFaceColor',[0 1 0]);hold on;
 scatter3(Y,X,Z,'MarkerFaceColor',[255/255 140/255 0/255]);
 xlim([0 250]),ylim([0 250]);
 axis('square');%view(-30,20);
 view(2);
 xlabel('C_{b}','FontSize',15);ylabel('C_{r}','FontSize',15);zlabel('Count','FontSize',15);
```

Question – 2

```
%Reading in the image
```

```
img_1=imgaussfilt(imread('Holi1.jpg'),1.5);
figure, imshow(img_1,[]);
title('Original holi image');
fprintf('Now select the region of interest');
cropped_img=imcrop(img_1);
figure, imshow(cropped_img);
[bar_conc norm_bar_conc]=hist_conc(cropped_img);

%Separating out the three channels in the image:
r_img=cropped_img(:,:,1);
figure, imshow(r_img);
g_img=cropped_img(:,:,2);
figure, imshow(g_img);
b_img=cropped_img(:,:,3);
figure, imshow(b_img);

figure, bar(bar_conc,'Barwidth',1);
figure, bar(norm_bar_conc,'Barwidth',1);

%Verifying that the normalized histogram sums to 1:
sum_hist=sum(sum(norm_bar_conc))

%%
%Read the strawberry image:
berry=imread('Strawberry.png');
fprintf('Now select five regions from the image');
berry_1=imcrop(berry);
berry_2=imcrop(berry);
berry_3=imcrop(berry);
berry_4=imcrop(berry);
berry_5=imcrop(berry);
[hist_1 norm_hist1]=hist_conc(berry_1);
[hist_2 norm_hist2]=hist_conc(berry_2);
[hist_3 norm_hist3]=hist_conc(berry_3);
[hist_4 norm_hist4]=hist_conc(berry_4);
[hist_5 norm_hist5]=hist_conc(berry_5);

figure, bar(hist_1,'Barwidth',1);
figure, bar(norm_hist1,'Barwidth',1);

figure, bar(hist_2,'Barwidth',1);
figure, bar(norm_hist2,'Barwidth',1);

figure, bar(hist_3,'Barwidth',1);
figure, bar(norm_hist3,'Barwidth',1);

figure, bar(hist_4,'Barwidth',1);
figure, bar(norm_hist4,'Barwidth',1);

figure, bar(hist_5,'Barwidth',1);
figure, bar(norm_hist5,'Barwidth',1);

%Computing distance between a pair of normalized histograms:
norm_hist_cell={norm_hist1,norm_hist2,norm_hist3,norm_hist4,norm_hist5};
num_hist=5;
distance_norm=zeros;
```

```
for i=1:num_hist
   for j=1:num_hist
      distance_norm(i,j)=dist_norm(norm_hist_cell{i},norm_hist_cell{j});
   end
end

%Annotation Routine

line([1 32],[max(norm_hist5)/2 max(norm_hist5)/2]);hold on;

line([33 64],[max(norm_hist5)/2 max(norm_hist5)/2]);hold on;

line([65 96],[max(norm_hist5)/2 max(norm_hist5)/2]);

text(1,max(norm_hist5)/2,'\mid');

text(32,max(norm_hist5)/2,'\mid');

text(15,max(norm_hist5)/1.9,'Red');

text(15,max(norm_hist5)/2.2,'32 bins');

text(0,max(norm_hist5)/1.9,'0');

text(28,max(norm_hist5)/1.9,'255');

text(33,max(norm_hist5)/2,'\mid');

text(64,max(norm_hist5)/2,'\mid');

text(45,max(norm_hist5)/1.9,'Green');

text(45,max(norm_hist5)/2.2,'32 bins');

text(33,max(norm_hist5)/1.9,'0');

text(60,max(norm_hist5)/1.9,'255');

text(65,max(norm_hist5)/2,'\mid');

text(96,max(norm_hist5)/2,'\mid');

text(75,max(norm_hist5)/1.9,'Blue');

text(75,max(norm_hist5)/2.2,'32 bins');

text(65,max(norm_hist5)/1.9,'0');

text(96,max(norm_hist5)/1.9,'255');

function [bar_conc norm_bar_conc]=hist_conc(cropped_img)

%Separating out the three channels in the image:
r_img=cropped_img(:,:,1);
%figure, imshow(r_img);
g_img=cropped_img(:,:,2);
```

```
%figure, imshow(g_img);
b_img=cropped_img(:,:,3);
%figure, imshow(b_img);

%Histogram for the red channel:
[r c]=size(r_img);
%stringing out the red channel as a vector:
r_resh_img=reshape(r_img,1,(r*c));
g_resh_img=reshape(g_img,1,(r*c));
b_resh_img=reshape(b_img,1,(r*c));
num_bins=32;
num_levels=256;
bin_size=num_levels/num_bins;
counts_red=zeros;
counts_green=zeros;
counts_blue=zeros;
n=zeros;
for i = 1:bin_size:(num_levels-7)
    n(i)=i;
   counts_red(i)=numel(find(r_resh_img((r_resh_img>=(i-1))&(r_resh_img<(i+6)))));
   counts_green(i)=numel(find(g_resh_img((g_resh_img>=(i-1))&(g_resh_img<(i+6)))));
   counts_blue(i)=numel(find(b_resh_img((b_resh_img>=(i-1))&(b_resh_img<(i+6)))));
end
count_red=counts_red(1:bin_size:length(counts_red));
count_green=counts_green(1:bin_size:length(counts_green));
count_blue=counts_blue(1:bin_size:length(counts_blue));
bar_conc=[count_red count_green count_blue];
total_val=sum(count_red+ count_green+ count_blue);
norm_bar_conc=bar_conc./total_val;

end
```

Question -3

```
zoneplate = imzoneplate; %generate the zoneplate image using imzoneplate.m by Steve Eddins

imwrite(zoneplate,'zoneplate.png');

zoneplatepng = imread('zoneplate.png');

imwrite(zoneplate,'zoneplatejpeg.jpeg');

zoneplatejpeg = imread('zoneplatejpeg.jpeg');

%imshow(zoneplatepng);

%imshow(zoneplatejpeg);

plot(zoneplate(2,:));xlim([0 500]);axis('square');%plot the intensity profile of zoneplate

zpng1d = reshape(zoneplatepng,1,501*501);

zjpeg1d = reshape(zoneplatejpeg,1,501*501);

figure, histfit(double(zpng1d),128);xlim([0 250]);axis('square');title('Color histogram for .png file');
```

```
figure, histfit(double(zjpeg1d),128);xlim([0 250]);axis('square');title('Color histogram for .jpeg file');
imwrite(zoneplate,'zoneplatejpeg25.jpg','Quality',25);
zoneplatejpeg25 = imread('zoneplatejpeg25.jpg');
imwrite(zoneplate,'zoneplatejpeg100.jpg','Quality',100);
zoneplatejpeg100 = imread('zoneplatejpeg100.jpg');
zjpeg251d = reshape(zoneplatejpeg25,1,501*501);
zjpeg1001d = reshape(zoneplatejpeg100,1,501*501);
subplot(1,4,1);
histfit(double(zpng1d),128),xlim([0 255]),title('PNG image');
subplot(1,4,2);
histfit(double(zjpeg1001d),128);xlim([0 255]);title('Quality 100');
subplot(1,4,3);
histfit(double(zjpeg1d),128);xlim([0 255]);title('Quality 75');
subplot(1,4,4);
histfit(double(zjpeg251d),128);xlim([0 255]);title('Quality 25');
 %difference image
%imshowpair(zoneplatepng,zoneplatejpeg,'diff');
```