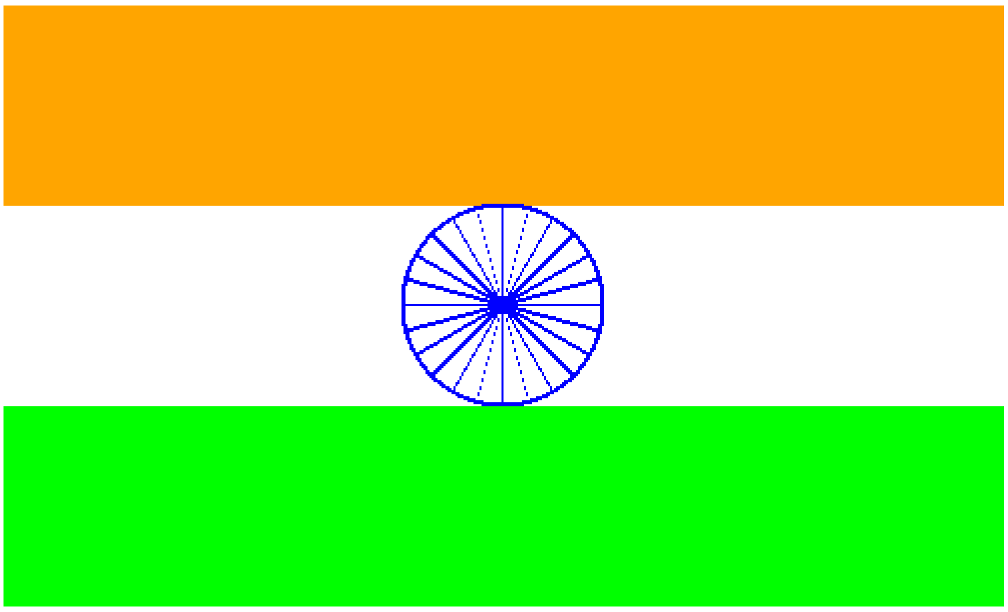
<u>CV ASSIGNMENT 1 REPORT</u>

The pseudo-codes of the questions are:

Question 1:

1. A numpy array of size 300 * 500 * 3 is taken.
2. Now the 3 horizontal bands of orange, white and green are drawn by assigning the appropriate RGB values of the colors in the numpy matrix.
3. Next the co-ordinates of the circle is taken in the exact middle of the white band.
4. A circle of blue color is drawn with a radius equal to 50.
5. Now one horizontal line and a vertical line passing through the center is drawn in the circle.
6. Thus 4 spokes have been drawn in the circle and 20 more need to be drawn in the circle.
7. So, 5 spokes are drawn in each quadrant of the circle such that they make an angle of pi/12 with each other.
8. The 5 spokes in each quadrant are drawn such that each point (x, y) on the spoke satisfies the condition that:
   If the center of the circle is (m, n)
   y – n = tan(angle) * (x - m)
   where the angles are in the range pi/12 to 23 * pi / 12
9. In this way the 3-dimensional numpy array for the flag is made and it has the RGB values of each of the pixels.

Image generated:



<u>Question 2:</u>

1. Here first I have created 2 folders: one containing images of car and the other containing images of dog.
2. One by one image is read from the folders and a HOG vector is created of each image.
3. To create HOG vector, first of all the image is divided into 8 * 8 cells and for each cell the histogram is prepared.
4. To create the histogram of the cell, the magnitude and gradient values are calculated for each of the pixels in the cell. The histogram of the 8 * 8 cell is a 9 * 1 size vector.
5. After calculating the histogram of each cell, we consider 16 * 16 patches. Thus, each patch consists of four 8 * 8 cells.
6. Now the feature vector of the patch is prepared by concatenating 4 vectors. So, the 36 * 1 size vector of the patch is made after normalizing the values by dividing each of them by square-root of sum of squares of the 36 values.
7. In the end the feature vector of the image is prepared by concatenating the feature vectors of all patches.
8. So, now I have the feature vectors of the 24 images that I have taken and their original tags i.e. car or dog.
9. This data is split into train and test data in ratio of 70:30
10. Now, the supervised ML model is trained. Here, I have used KNN.
11. The labels are predicted for the test set using the model and accuracy is calculated. The accuracy of the ML model comes out to be 75.0% over here.

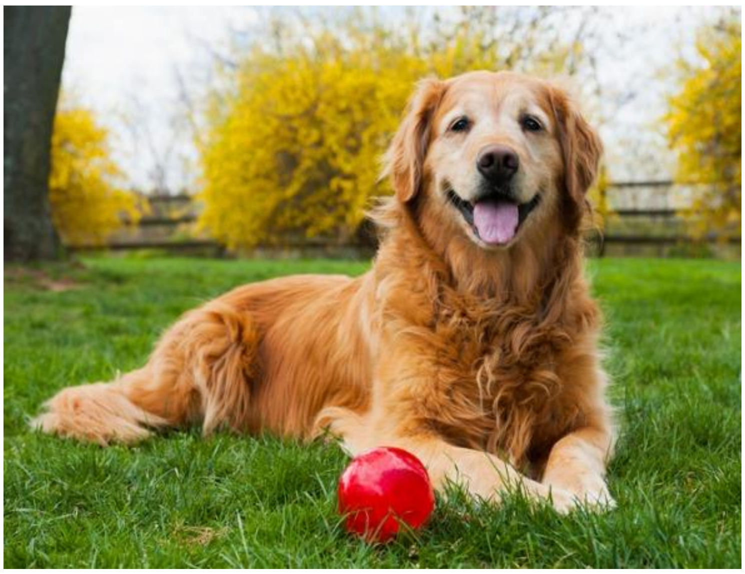
<u>Question 3:</u>

1. Here I have taken an image of a dog with some background. I have created a second image by changing the background to solid blue color. Also, the position of the dog in the second image has been changed. So, now I have 2 images and they are now used to find the matching patches.
2. Each of the image is resize to a dimension of 256 * 256. Also, the image is converted from RGB to gray scale.
3. In the 1st image patches of size 32 * 32 are selected and the LBP values of each of the pixels in the patch are calculated. Now a histogram of the patch is obtained based on the LBP values. This histogram is a 256-

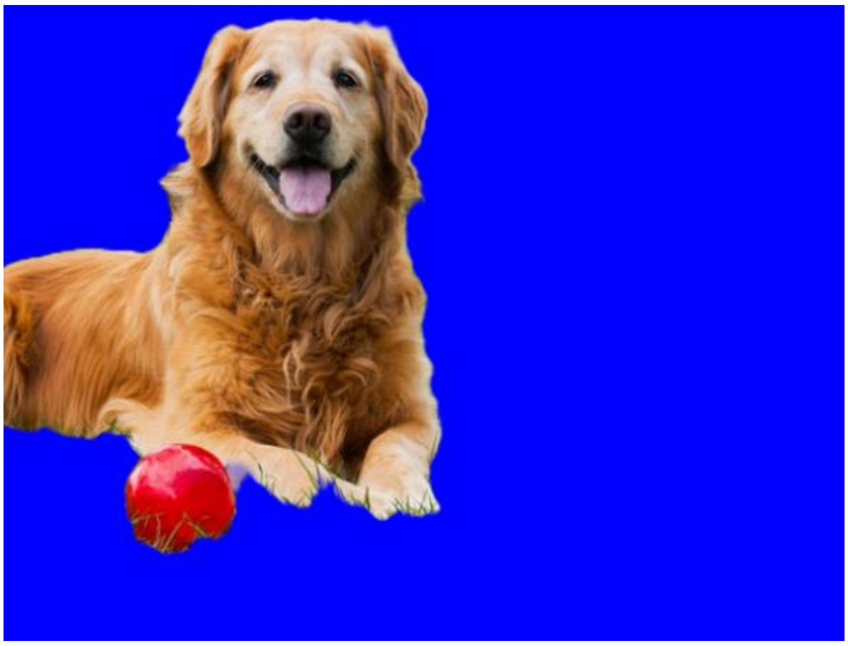
dimensional vector as the LBP values range from 0 to 255. Also, the vector is normalized by dividing each value by the sum of values in all dimensions.

4. Similarly, the vectors for 2nd image are obtained.
5. Now the process of matching of patches is done.
6. Here, I have used the chi-square distance between the histograms of the patches as a measure of the similarity of patches.
7. One by one a patch is picked from 1st image and the chi-square distance is calculated with each patch of 2nd image. If the chi-square distance between the corresponding histograms is zero then the patches are matched.
8. Now the matching patch ids are displayed which are the patch serial numbers when patch creation starts from top left of image.
9. Also, I have extracted the corresponding patches from the RGB matrix of 1st image and stored all the patches images in the folder named Patches.

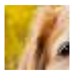
The image that I have taken as input is:



Now for the above image I have changed the background and also changed the location of the object in the image, to get the below image:



Now, using LBP features on finding the matching patches, some of the matching patches are shown below:

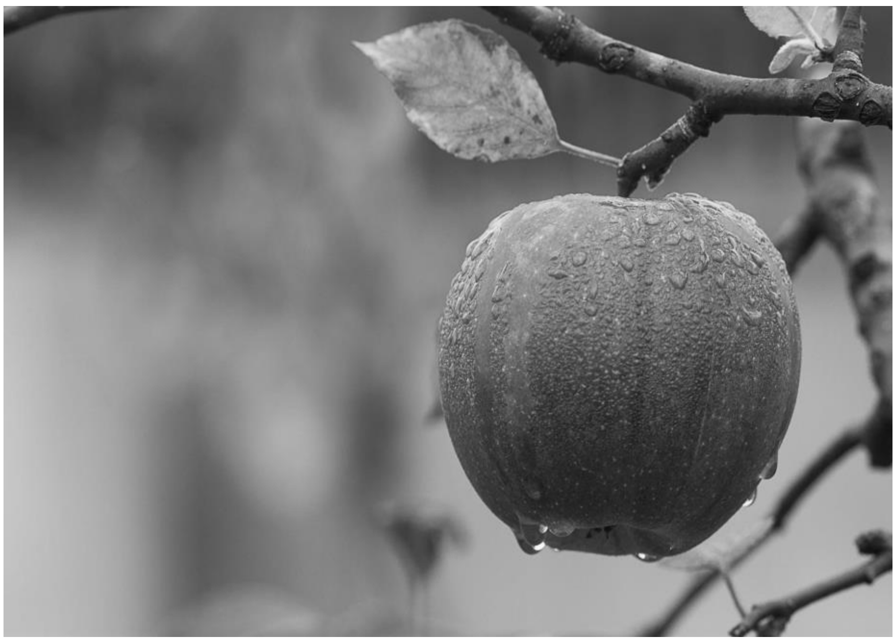


So, it can be seen that features of the dog which are common in both images are matched.

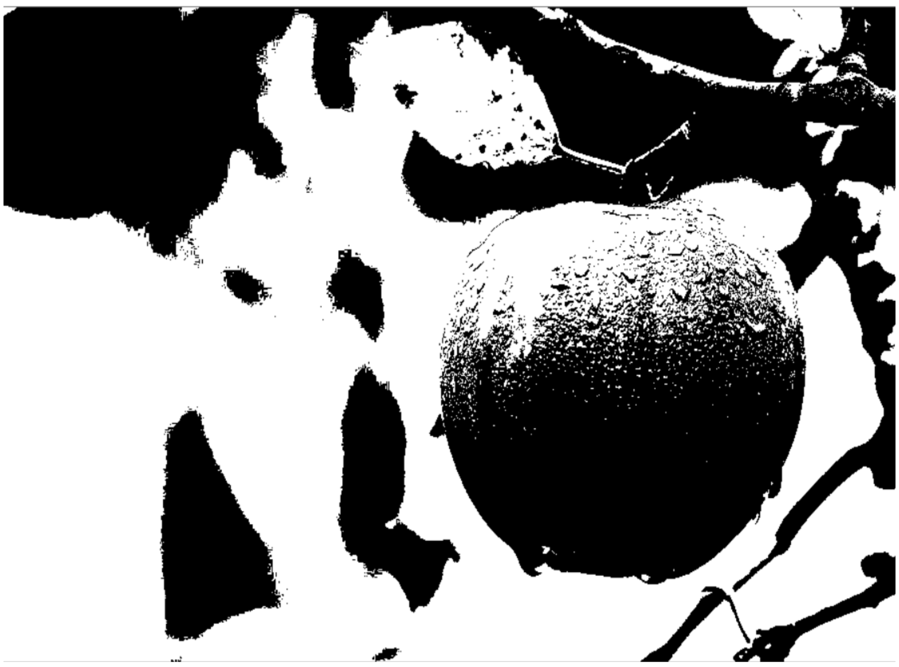
<u>Question 4:</u>

1. A gray scale image is taken as input.
2. Now the histogram of the intensity values of the pixels of the image is generated.
3. Over here there are 2 classes i.e. background and foreground.
4. By considering different values of threshold, the within class variance is calculated.
5. The threshold intensity value is to be calculated in such a way that the within class variance is minimized.
6. The variance is calculated using the variance and threshold is found.
7. All pixels having intensity values less than or equal to the threshold are background and are assigned intensity value = 0 and those with intensity values greater than threshold are foreground and assigned intensity value = 1.

Input image:



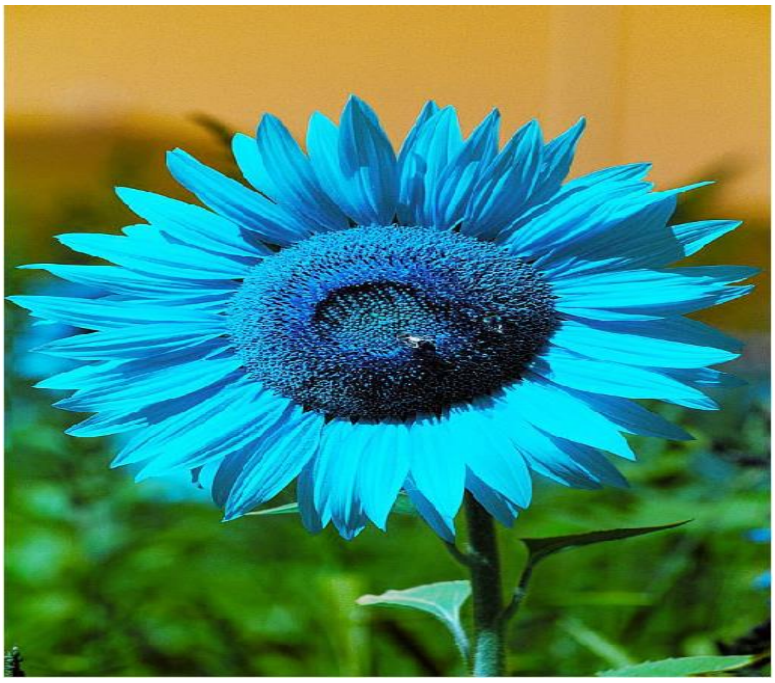
Output image:



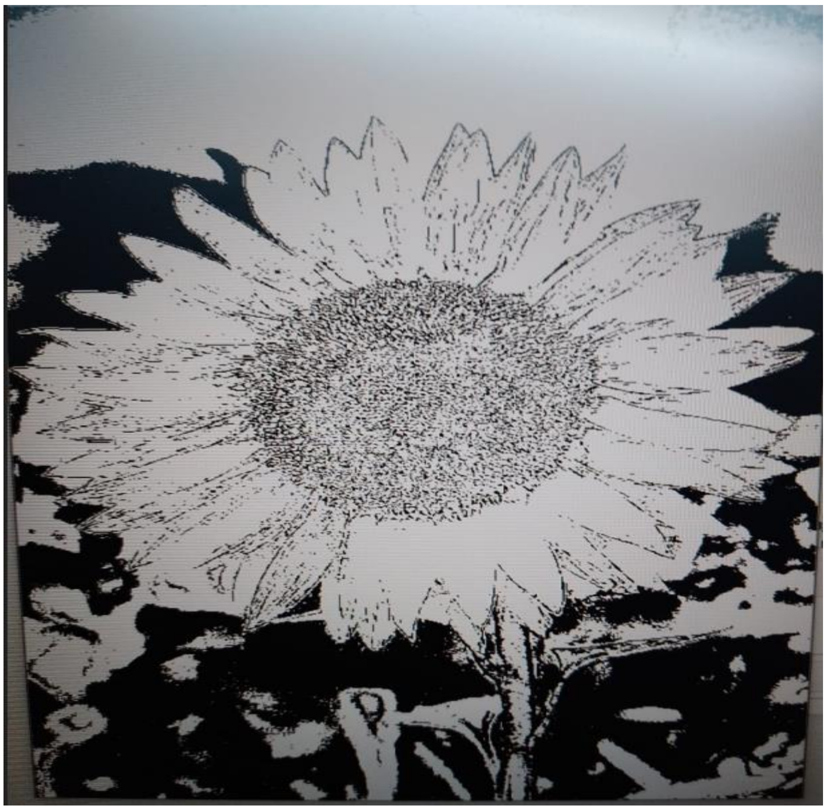
Question 5:

1. Here initially k pixels are randomly selected as centroids of clusters.
2. Now for each pixel of the image, the Euclidian distance between the pixel RGB values and the RGB values of the cluster centroids are calculated. The pixel is assigned to the nearest cluster.
3. In the next iteration the centroids of the clusters are found again by finding the mean of the respective RGB values of pixels. Each pixel is reassigned to a cluster by finding the centroid with minimum Euclidian distance.
4. The above process is repeated till Euclidian distance between the centroids at previous iteration and the new centroids is less than a threshold which I have taken to be one.
5. In the end the image is segmented into k clusters and it is displayed.

Input image



Output Image:



Question 6:

1. Here I have taken as input an image which has circles.
2. The image is resized to dimension of 256 * 256 and the image is converted to gray scale.
3. Now the edges in the image are developed using the ImageFilter of the PIL library of python.
4. A binary image is generated by applying thresholding.
5. Also, the following is true about a circle:
   A circle can be described in terms of the co-ordinates of its center i.e. (a, b) and the radius of the circle i.e. ‘R’.
   Now,
   $x = a + R\cos\theta$
   $y = b + R\sin\theta$
   These eqns. Can be used to draw the circle i.e. (x, y) are the points through which circle passes.
   When θ varies from 0 to 360 degrees the complete circle is generated.
6. So, for every pixel with co-ordinates (x1, y1) that is a part of edge i.e. its pixel value is 255, the following are calculated:
   $a = x1 – R\cos\theta$

$b = y1 - R\sin\theta$

Here R is varied from some min value to minimum(width/2, height/2).
For each R value, θ is varied from 0 to 360 degrees.

So, values for each triple (a,b,R) are maintained which denote the no. of circle generated with the above parameters.
The triples with the greater number of counts are the circles present in the image.

Here the time complexity of the algo is too high, due to computational constraints the algo. Was taking too much time to run.
Although, I have coded the complete algorithm.