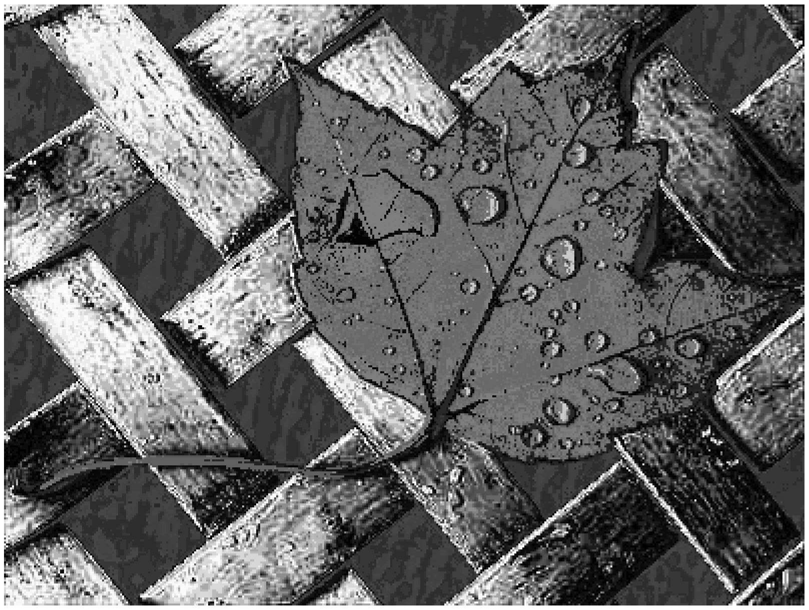
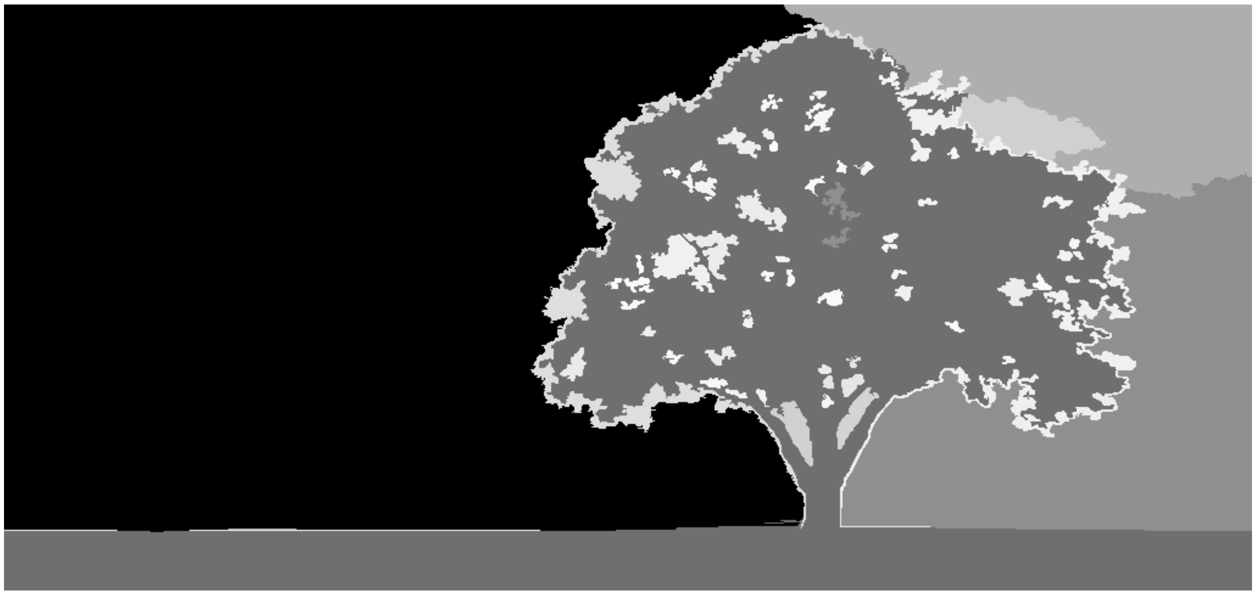
Q1
Part 1:
First use the matlab code provided in class to convert the image given(leaf.png) to image with 85 colours(using kmean algo) saved as leaf85.png(provided in Data/Input folder)
Now read this leaf85.png(stored in a variable named im).using numpy.unique found all the unique colours and the count of them. Defined function to find the distance between colours.
Normalised the frequency of the colours(as stated in paper)(stored in variable named freq)
For every distinct colour found the saliency value using the **equation 3** provided.
Brought the values in the range of 0-255(so that they can be stored as an image).
Created a new image with the saliency value, reshaped the flatterend variable so that it takes the shape of a image, then saved it as ouput1Q1.png(attached below)

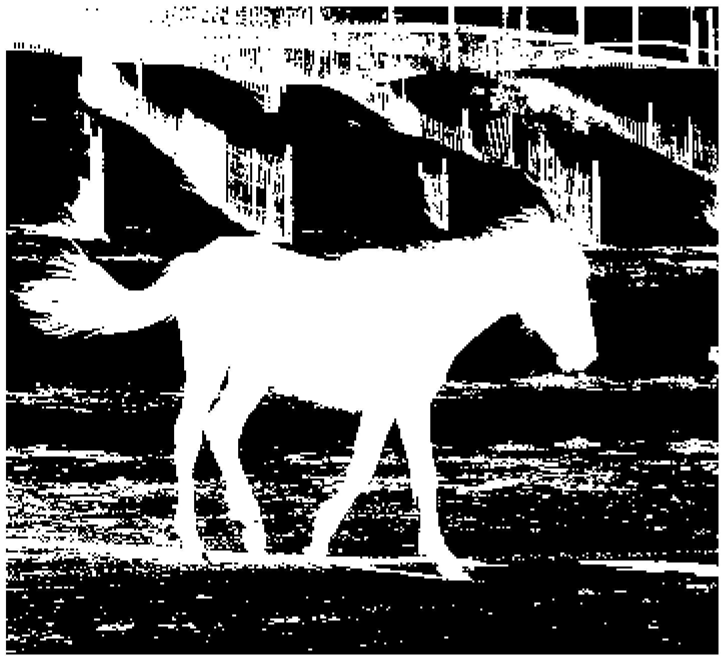


Part 2:
First, I needed to get the regional proposals. To get this used the code provided. Unzip the file given, make some changes in the main.py(change path to image, and save the final output as image(`seg_tree.png`)).
Read the original image(BigTree.png in variable im) and the region proposed image (seg_tree..png in variable im_seg). Found the distinct colour region proposed(again using np.unique).
Stored all the pixels of the original image according to their regions(in variable REGIONS).
Again stored all the distinct colours and their frequency/probabilities in variable RDCLR and RFREQ respectively.
Defined a function which takes two regions(index) and gives the distance between them(implemented equation 6 of the paper).
Finally implemented **equation 5** and found saliency values for each region.
Again brought the saliency value to 0-255 range. Stored the saliency map as a image as the final output(named: `output2Q1.png`) (attached below)



Q2

Read the image(`horse.jpg`) in grayscale(stored in variable img). Flattened it(img_flat).
Created histogram from the flatten image(in 255 bins) and then normalized the values.
Defined a function of the total sum of squares.
For all 255 values of threshold possible, found the tss of both fractions and compared the values to get the threshold with min value.(also storing the data in variable data_csv)
Saving the tss values for each threshold value in a csv file named `th_file_Q2.csv.`Final row contains the final threshold value.
It can be found in the Data/Output folder.
Using the final threshold value for creating a binary mask, use this mask to make an image.
Stored again in Data/Output folder named output2.png(attached below)



Q3:
Input: pass path to the folder containing the video as input example:
M_2019423_A1_Q3('D:\learn\SEM6\CV\A1\final\Q3\')

The video named shahar_walk.avi is read and stored in variable vid.
The video is read frame by frame and frames are stored as image in same folder
Now read the frames and store them in a variable base. Whose 4th dimension stores different frames.
Take median of this variable(pass dimension so that median is taken as pixel wise)
Now read each frame, subtract the base from it.
Now decide a threshold value.
Get the bounding box value, use this value to plot the rectangle. Made the curvature as [1 1] so that it becomes a ellipse(around the object)
Now save this to the video file

The final output is provided in the Data/Output/output3.avi file.
The code provided is well commented for understanding.

ALL THE CODE PROVIDED IS WELL COMMENTED(in case of any query please have a look at the code)