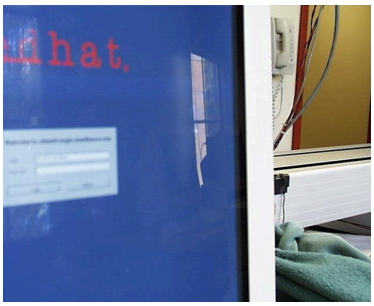
Content Based Image Retrieval

In this Project, I have implemented content based Image retrieval based on different feature matching algorithms. Below are the different algorithms implemented. All the images are read and the csv files are read using the methods provided by the professor.

1. **Baseline Matching:** In this method, for each image the central 7*7 pixel values are taken and stored as a series of values in a csv file. So, the csv file will contain all the images feature values. In the second step, the csv file is read and the target image is identified. If the target image is not present, an error is thrown. The target image feature vector is taken and then distance is calculated w.r.t all other images vectors in the database using the sum of squared differences as a distance metric. Then the top image input in the command line is output as results.
   For each implementation, there are two steps, first is to store the feature vector of images in the file and the second part is to read and find the scores w.r.t to distance to target image and get the best ranked images.
   For all the implementations, we have made storing and ranking of images method common and we pass specific functions according to the kind of implementation being done.
   Below are the results of the Baseline matching implementation.

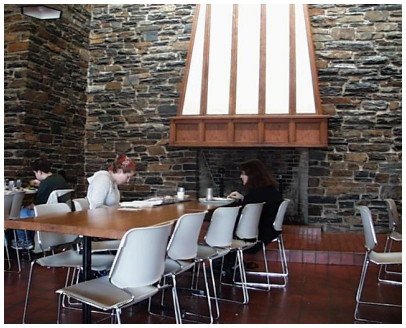

2. **Histogram Matching:** In this implementation as well, we have two parts , first is to store all the images feature vectors. Here, I have considered the RG Chromaticity histogram. I have considered 16 bins and this can be customized as required. The results shown here for 16 bins. To find RG Chromaticity, we have considered a 2d vector with 16 bins. The r value is taken as red/(red+green+blue) and g value is taken as green/(red+green+blue). The bin value is calculated as (int)(r_value*(numBins -1)+0.5). 0.5 is added to make the histogram more even and unbiased. The histogram scores are saved as the vector values in the csv file. Now, the target image vector is compared with all images using histogram intersection as distance metric. This takes the common intersection between the two vectors and the higher value of intersection indicates more common features and hence more similar the images are.
   Below are the top 10 results for target image pic.0164.jpg:

```
(base) sachinpc@Sachins-MacBook-Pro build % ./Project2 pic.0164.jpg rank_rghistogram_m 10
Reading /Users/sachinpc/Documents/Northeastern University/Semesters/Fourth Semester/PRCV/Projects/2/utilities/featureFiles/rgChromaticityFeaturesFile.csv
Finished reading CSV file
Target Image name = pic.0164.jpg
totalFiles = 1121
1120. pic.0164.jpg
1119. pic.0080.jpg
1118. pic.0489.jpg
1117. pic.0461.jpg
1116. pic.0898.jpg
1115. pic.0486.jpg
1114. pic.0110.jpg
1113. pic.0426.jpg
1112. pic.0166.jpg
1111. pic.0923.jpg
1110. pic.0292.jpg
Terminating
```

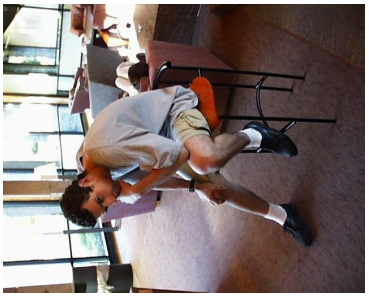
3. **Multi Histogram Matching:** For this implantation, I have considered using rgChromaticity histogram for the upper half of the image and lower half of the image. The main rationale is that, here, its trying to match the chromaticity specifically w.r.t upper and lower images separately and hence, we might get matching similar to the target image.The process of getting rgChromaticity is similar to previous method and only half of the image is passed.
   Below are the top results for target image pic.0274.jpg:

4. **Texture and Color Matching:** For color matching, I have considered rgChromaticity for the whole image and for texture matching I have considered sobel magnitude and have created a histogram for the values of Sobel M as texture. So, the features are the combination of color and texture values in histogram. Now, both are of equal weight and have compared the target scholar features with other images color features and target texture features with other images texture features. Now, rgChromaticity and the magnitude common area is calculated as distance metric and summed to get the score. Higher the score, better the matching.
   Below are the top results for target image pic.0535.jpg:



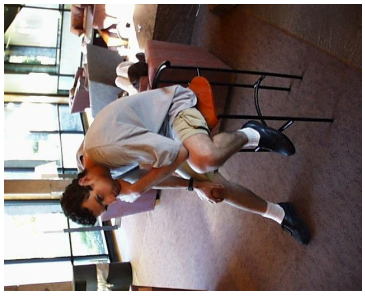

For the same image, below are the top results using single color Histogram. (EXCLUDING TARGET IMAGE).





Below are the results using two color Histograms(EXCLUDING TARGET IMAGE).



We can see from the above results, the top images obtained using color and feature vectors have images which have similarity in color and also edges. If we see the images, they also have similar edge structure as the target image. Whereas, the single and two color histogram images results are based on color and we can see the images are of similar colors of target image. We can also see some common images in all three results because of the high similarity in color. The two color histogram images are considered based on the top and bottom half colors of the target image and can see that results are similar to the upper and lower half of the target image.

5. **Deep Network Embeddings:**

Below are the top results for target image pic.0893.jpg:



Below are the top results for target image pic.0164.jpg:

To compare, lets see the results for image pic.0893.jpg using single color histogram

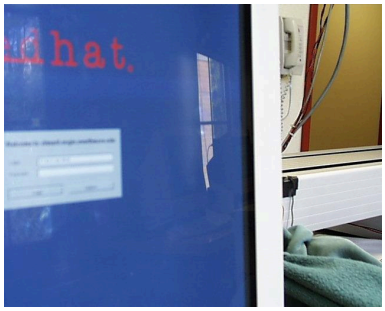
Results of applying a single color histogram method to pic.0164.jpg are already given above.

The results for image pic.0893.jpg using two color histograms method:




Below are the top results for target image pic.0164.jpg using two color histogram:





The results for image pic.0893.jpg using texture-color histogram method:

The results for image pic.0164.jpg using texture-color histogram method:

We can clearly see from the results that Deep network embeddings look into the features very thoroughly and hence the results are more accurate compared to using single color or two color histograms. The next two methods focus majorly on color and hence does not consider other aspects of the image which is not the case with Deep network embeddings features. So, the results w.r.t color histograms are more oriented towards matching color of the target images and we can see that the results have similar colors to that of the target image. The texture-color histogram feature also considers edges as texture and hence we can see that for image pic.089r, the resultant images are similar in

edges and also color aspect of the target image. We can also see that it correctly identifies an image exactly similar to the target image because it matches with the edges and also the color features of the target image.

6. Analysis of Results

Lets consider Image 1072:

Results of Deep network Embeddings:

Results of Single Color Histogram:




Results of Two Color Histogram:




We can see here that, all the images are majorly color intensive and hence, we can observe that the results on Single color and two color histograms are on par with the results of Deep network embeddings method. When the color is a major feature in the target image, then the color histograms method does really well in identifying the images similar to target images. The Deep network embedding also performs well as it learns the different features of the target image thoroughly.

Results for using Color texture Feature:




Here also, we can see that most of the top images are predicted very well. However, its performance is not that good compared to using only color histogram because, here edges as texture is not that important and hence considering it might give us some bad results as image 3 which also gives some weightage to the edge texture which is not a good feature for this image.

Now, lets consider image 0948:

Results of Deep network Embeddings:

Results of using Single Color Histogram:



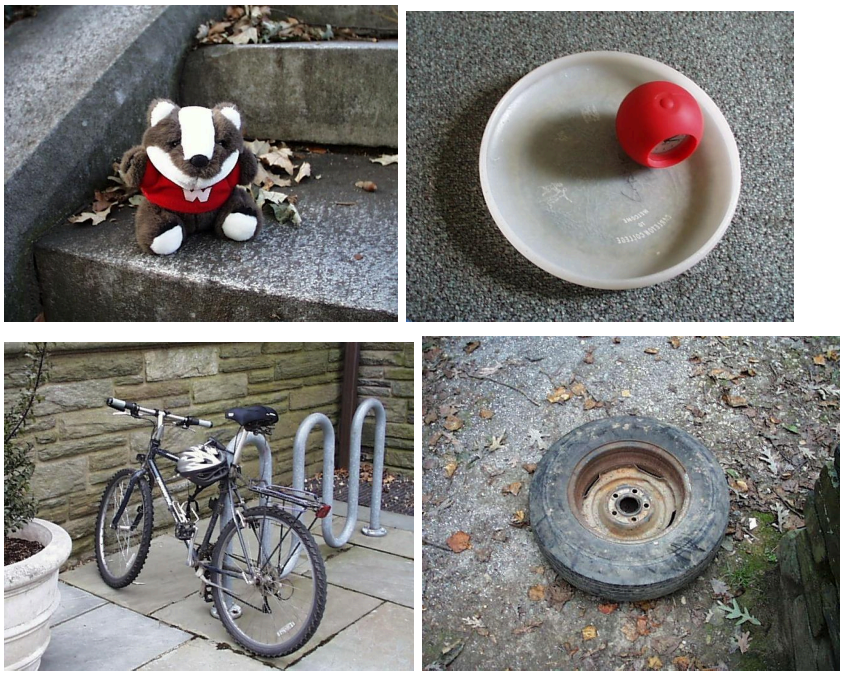

Results of using Two Color Histograms:




We can see from this image that the Deep network embedding features method works really well in identifying the relevant features of the target. However, the color histograms

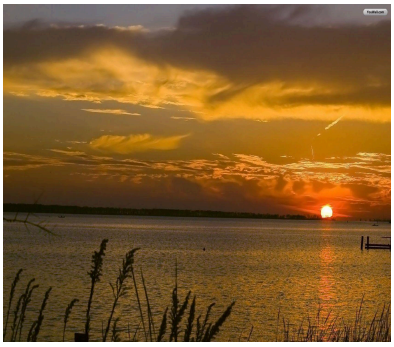
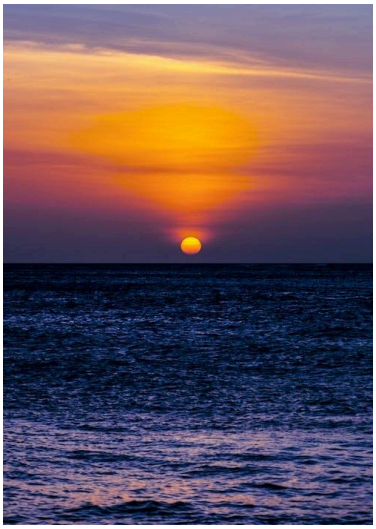
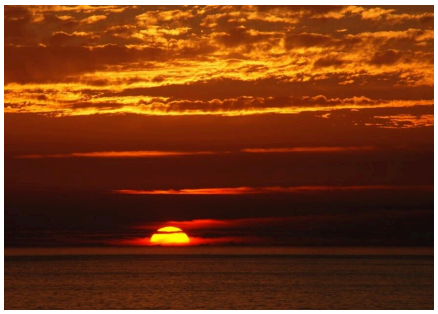
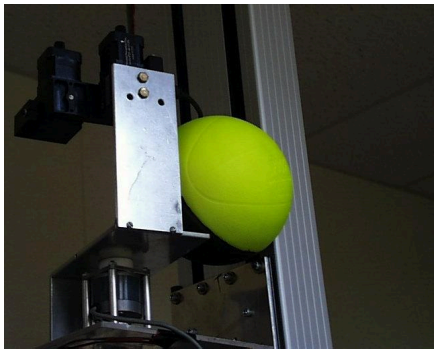
are very bad. Because, it just picks colors and clearly, in the target image, color is not the main feature. The edge detection might have been a good approach here.

Results of using Color and Texture feature:



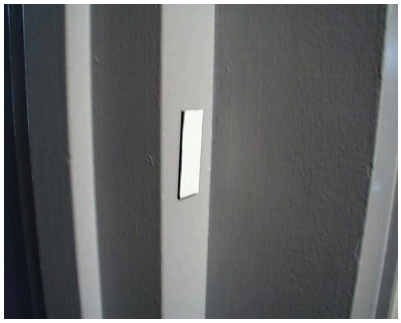
This method also does not perform that well compared to Deep Network embedding method because here equal weightage is given to both color and texture Sobel M and hence, may be because of that it is unable to only focus on the edges and hence the results are not accurate.

7. **Custom Design:** In this, I have chosen to detect images with sunsets. The idea I have used here is that the color around the sun is usually very bright and hence we can use the HSV color model to represent the image to detect the intensity and colors of the images. The given RGB image is converted to HSV image so as to better identify the intensity of colors in the given image. The images with sum images will have a hue value between 0 to 30 and the intensity and color values are high. Hence, we are thresholding the pixels with hue value between 0-30, saturation value between 100 to 255 and value between 100 - 255. These are the values which are tested to best identify images with sunsets. We normalize the number of pixels contained in the image which meets the threshold by the overall pixel values. All the images are ranked on this score and the ones with higher scores represent the images with high intensity color images which have colors similar to the intensity of the setting sun. Below are the results obtained.


Here we can see that, w.r.t color and intensity of the image it is able to identify the images with the color and intensity similar to the sun. It is able to identify sunsets as well as some other images which contain color intensities similar to the sun or lie in the hue range of 0-30, saturation range of 100 -255 and value of 100 - 255. Fine Tuning these values might yield different/better results.

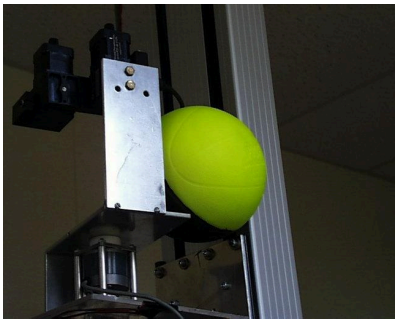
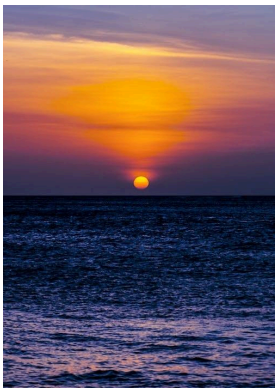
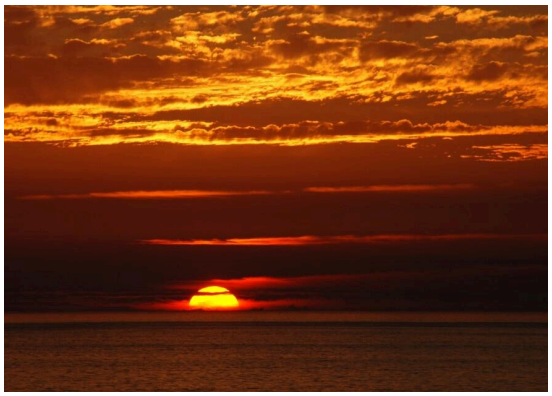
Below are the images with least matching to detecting sunsets


We can clearly see that these images have no similarity to sunsets and are quite opposite to that of color intensity w.r.t sun. Hence, these images are ranked very low.

8. **Extensions:** In the extensions I have tried to extend the implementation of detecting images with sunsets. There are two different types of implementations I have done for this.
    a. **Intensity of the pixels above threshold:**
    We know that, whenever there is a sunset in the image, there is a gradual increase in the intensity of the colors greater than the threshold considered above. So, in this, I have considered the saturation histogram and of only the pixels for which the values are greater than the threshold considered above. So, if there is a gradual increase and decrease in the saturation values, then they are ranked higher compared to constant or only increasing and decreasing saturation values. The rationale behind this approach is that, the nature of images containing Sun is the increase and decrease in the intensity of pixel values of colors which lie in a given threshold.

Top results of this approach are:

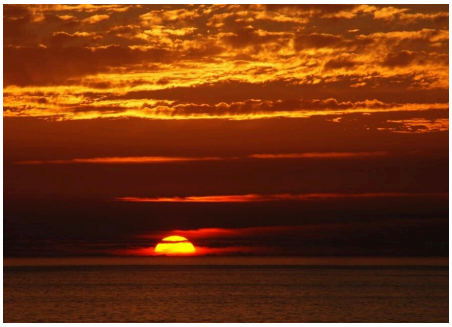

Here, only the intensity histogram is considered and considered the difference in increase in decrease of intensity of the threshold pixel values. We can see that it fairly does the job in identifying images with suns, however, its not perfect and still detects images with increase and decrease in the image threshold pixel values. One disadvantage of this approach is that it doesn't consider the continuity of the image color and if there is disoriented same color intensities, then also, it will identify as a potential image with sunsetting.

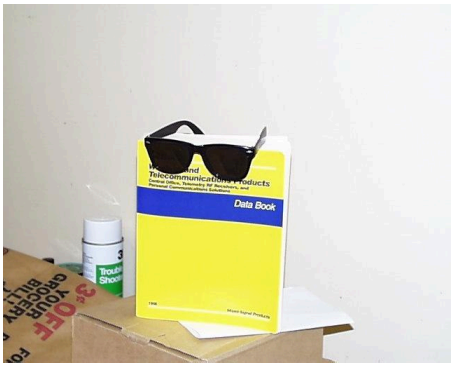
b. **Splitting Image row wise into n splits:**

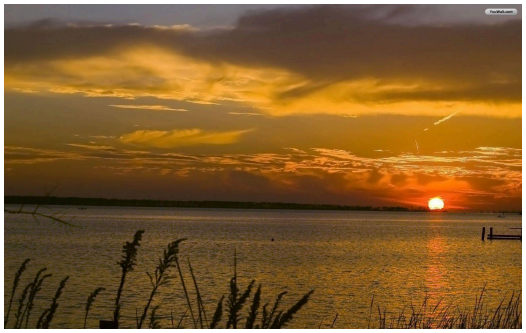
Splitting the given image row wise into n parts and for each party, calculating the pixels which lie in the given threshold and calculating the average intensity among all the pixels which lie in the given threshold. This is repeated for all n row splits of the image and for each split the normalized pixel score and the average intensity value is stored. Now, we know that each part of the image contains certain pixels Greater than the threshold and the intensity of those pixels for that

part of the image. So, now when the image is split into n different parts row wise, then with sunset images, one similarity will be identifying the pixelScore and the intensity of each part of the image. The images with the sun will have intensity of each part increasing and then reaches maximum value and then again starts decreasing. So, a scoring metric is created to consider the gradual increase and decrease in the intensity by finding the part of the image containing the maximum intensity.

The top results of the above implementation is given by:





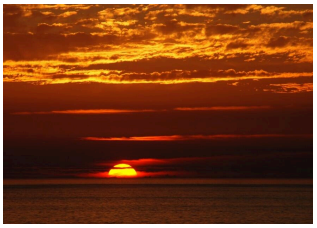


Here, we can see that the results consider an increase and decrease in the intensity based on the image parts row wise. Hence, we can see the top results also contain other images with similar behavior. There is still a lot to improve with this feature ang given more time and research, I can make it more optimized to detect sunsets.

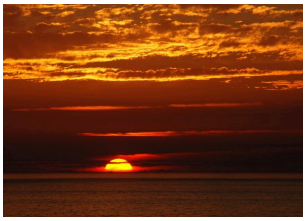
c. **Passing the target image with sunset and detecting as many images with sunsets:**

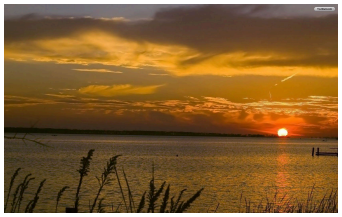
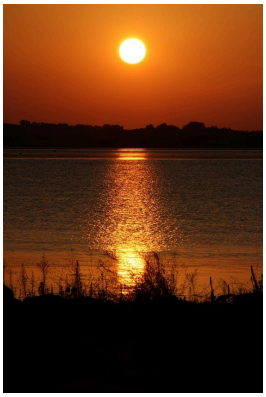
For this approach, we can extend on using the intensity histogram to identify the common area of histogram intersection to identify the images similar to image containing the sunset. Here the histogram contains the pixel intensities of pixels whose values are greater than a certain threshold. Hence, it will be a good feature to consider the intersection of it w.r.t the target image because the target image of setting sum will also have similar intensity ranges for the images consisting of sunset. There are a total 8 images in the database with sunsets.

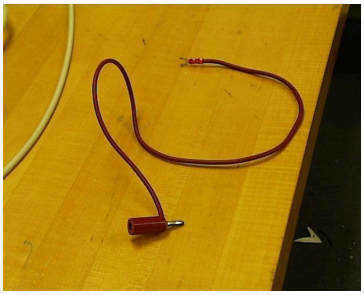
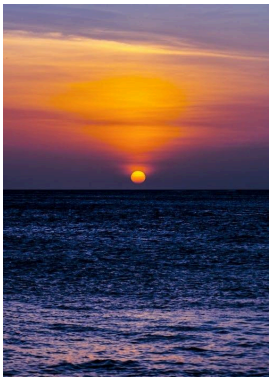
Target image considered is:



The results are:




We can see from the results that 5 of the top 10 images contain sunsets and the other images also have similar color intensity as the sun. Out of more than 1000 images, it is able to detect 5 sunset images in the top 10. This can be improved

by also considering the edges as well and identifying circular objects with the threshold pixel values. This will further increase the score. The disadvantage of this approach is, even if the threshold pixel values are not continuous which might not be the case in detecting the majority of sunsets in an image.