Project 2 – Content Based Image Retrieval

The purpose of this project is to co-relate/rank images based on the features of the target image to other images in the dataset. The features may include pixel color, texture or a bit of both. The program method adopted is method 1 (single program for each query). The project is divided into 5 tasks which are as follows –

TASK 1 – Baseline matching

This is a direct pixel to pixel RGB color intensity comparison between the target and the query images. This is implemented in the form of extracting 9X9 matrix from the center of the target image and comparing each pixel intensity (RGB) of target to query images. The distance metric used is least square metric. Following are the top 3 results for the *pic.1016.jpg as target.*

Fig. 1 – images ranked using baseline matching- (pic. 1016, 986, 641, 233 from left to right)

As we can see the red color dominates our target image and same can be seen in the ranked images.

TASK 2 – Histogram matching

Here instead of comparing a patch of our image, we compare rg chromaticity normalized histogram. For each pixel r and g chromaticity values are calculated and then put in 18X18 bins corresponding to their (r,g) chromaticity values. Then the values are normalized so that the sum of values in all bins is 1. Then for comparison histogram intersection method is used, where for (r,g) bins of target and query image we find min of (r,g) for each bin which is the intersection. Number closer to 1 means highly co-related. Hence we use 1 – distance as our metric. Following are the results for the *pic.0164.jpg* as target.

Fig. 2 – images ranked using rg histogram matching- (pic. 164, 80, 898, 489 from left to right)

We can confirm our results by visual inspection where all the images have similar proportion of blue and green.

TASK 3 – Multi-histogram matching

Using the same method as above, we have divided the target image into four sections and found histograms for each section and compared with the four sections as shown in first image of Fig. 3 of our query image.

This results in better results as each section is matched with its's corresponding section. Following are the results for the pic.0274.jpg as target.

Fig. 3 – images ranked using muti-sectional rg histogram matching- (pic. 274, 409, 825, 275 from left to right)

We can infer the results are visually more co-related than previous methods. We can also see that almost in all images the sectional distribution of colors is similar.

TASK 4 – Texture and Color matching

Here we are using color histogram and sobel magnitude histogram equally weighted as a comparison metric. Color histogram will signify the color distribution in our target and query image, while sobel magnitude will signify the boundaries texture in our image. We usewd histogram intersection matching as the distance metric to find our queries. Following are the results for pic.0535 as target image.

Fig. 4 – images ranked using sobel and color histogram matching- (pic. 535, 383, 480, 785 from left to right)

These results are even more impressive as we can see more or less equal distribution of color and similar quantity of texture (lot of boundaries because of sobel magnitude). The weights for color and texture are equal but if texture dominates in ranking it can dominate the color metric or vice versa.

TASK 5 – Custom Design

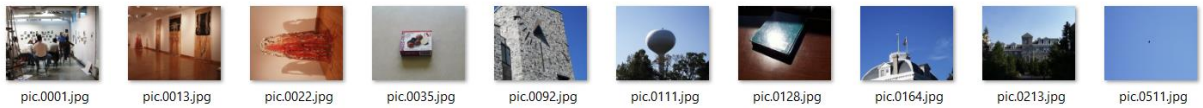
The custom metric we chose is to find sky in an image and then use a custom reward punish metric system to find the final decision metric for the query. We used HSV values for various sky images present in the dataset and calculated the optimum range of values they can be in for a blue sky. There can be a sky in query but with lots of clouds or it's inverted, such images won't be ranked in our list highly as we are searching for the HSV range of blue sky. For this function, The user just need to provide directory and it will rank the images with max blue sky present. The algorithm for this design is – extract the upper half of the image, then for each pixel find the reward/punish metric and move to the next metric.

To test our metric and design we have 4 sky images mixed with 6 without sky images (in easy(less blue), medium(blue present), hard(blue present substantially). The 4 expected top ranked images are as follows –

Fig. 5 – Observation ranking for our experiment (pic.511, 164, 111, 213)

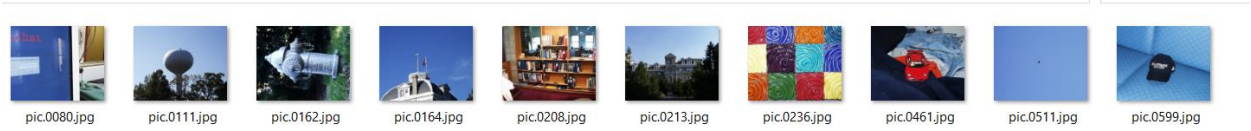
We calculate our performance as - % ((number of images lying in top 4 of above)/4 * 100) for each set. Following are the results for our custom design.

For easy set – **100%** all images were recognized and were ranked exactly as we jusdged visually. Snapshot of dataset attached below



For medium set - **100%** all images were recognized and were ranked exactly as we jusdged visually. Snapshot of dataset attached below. In medium, we made sure to add some % blue in the top section.

For difficult set – **75%** matching with a cap query spoiling the results because of blue background. The dataset for difficult must contain blue in the top but should not be sky.



Thus we can conclude that out method for comparison and metric is pretty strong for identifying blue sky in the image!! With an exception of completely blue upper half backgrounds!

EXTENSIONS –

We added two extensions to our project –

A. (easy) Make a collage with ranking in order to quickly visualize our results for query. The collage is saved in the executive file itself.
B. (Very difficult) Identifying banana in the center image!!

A. Collage

The collage is implemented to visualize a lot of images in the results in a go. The following is the result of performing Task 1, for 10 images on olympus dataset.

Fig. 6 – Extension A - Collage creation of 10 images for task 1.

B. Banana finder

To find the banana we have created a custom algorithm and metric. The algorithm is divided into six steps –

1. Locate the center portion as the area of search – For our train set, we have 5 bananas, we adjust the center portion dimensions so that all bananas are captured in the frame of inspection.
2. Apply Gaussian – A 5X5 gaussian blur is implemented on the query image as a primary texture identification.
3. Apply Sobel magnitude– For identifying the boundaries texture for banana.
4. Histogram matching for resultant sobel magnitude query with the reference image (banana in center); histogram matching for resultant gaussian query, and finding the distance metric (here using closer to 1 -> closer to reference)
5. HSV color picking in the extracted frame for banana. (Finding metric, normalizing metric).
6. Optimizing weights for different datasets for optimal results.

Following are the results for our analysis on easy, medium, and difficult set respectively->

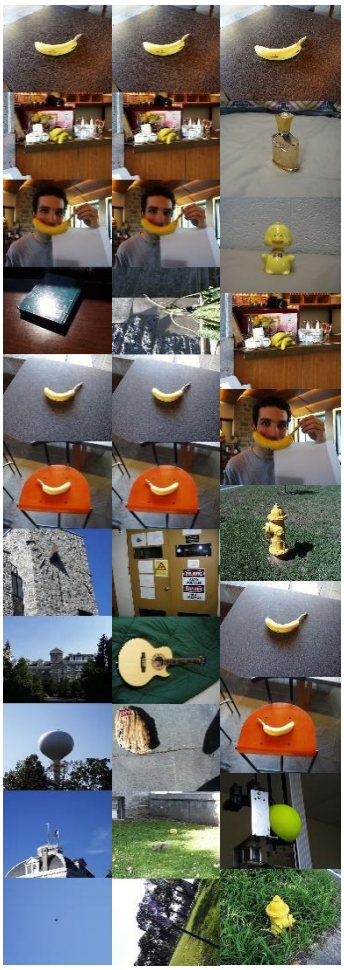


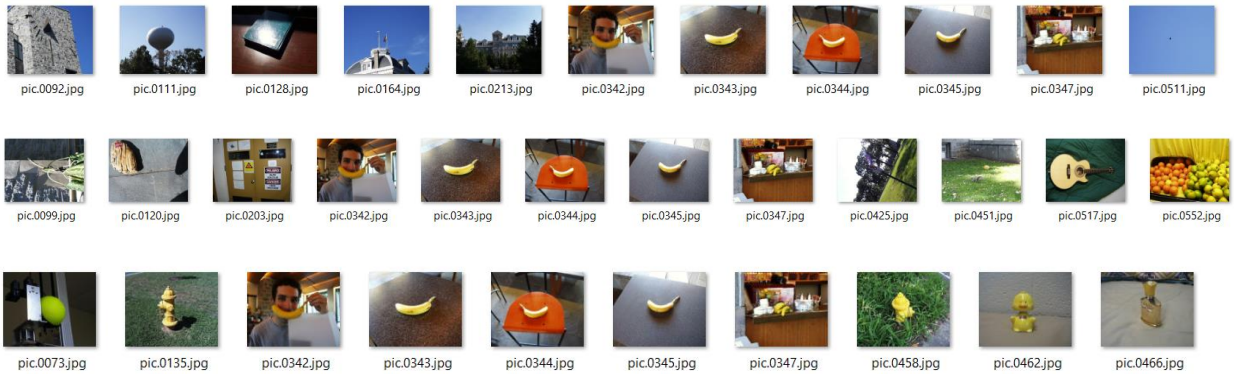
Fig. 7 – Extension B – Banana Finder

From the results, the percentage accuracy is 80%, 80%, and 60% for easy, medium and difficult set if we see the first 5 ranks for each image. The reasons for mismatch identified are –

1. When seen Sobel magnitude raw image for reference image there are lots of grains (noise) which can cause ranking issues.
2. Although matrix for comparison used in the process are sobel, gaussian, and color. All represent the color distribution and identifying shape of an object requires intricated algorithm currently out of scope.
3. Scale, rotation, invariance is not used as currently out of scope.

We could surely expect improvement in matching results when the above reasons are worked upon!

Names of jpg used for easy, med, diff set respective are –



Reflection –

This project started out as fun, but debugging took a lot of time. It was irritating at first but then systematically debugging helped. Disheartening to see images not matching at first but when they match its immense joy. Learned a lot of C++ in this assignment and surely feel more confident in the language. Overall, a fun learn ride. Each task reflection/conclusion is written in its section.

Acknowledgement –

Only used professor's lecture, OpenCV docs, and Stack Overflow for debugging.