CS5330 - Pattern Recognition and Computer Vision
(sec-01 spring 2023)

Northeastern University

Project - 2
Content Based Image Retrieval

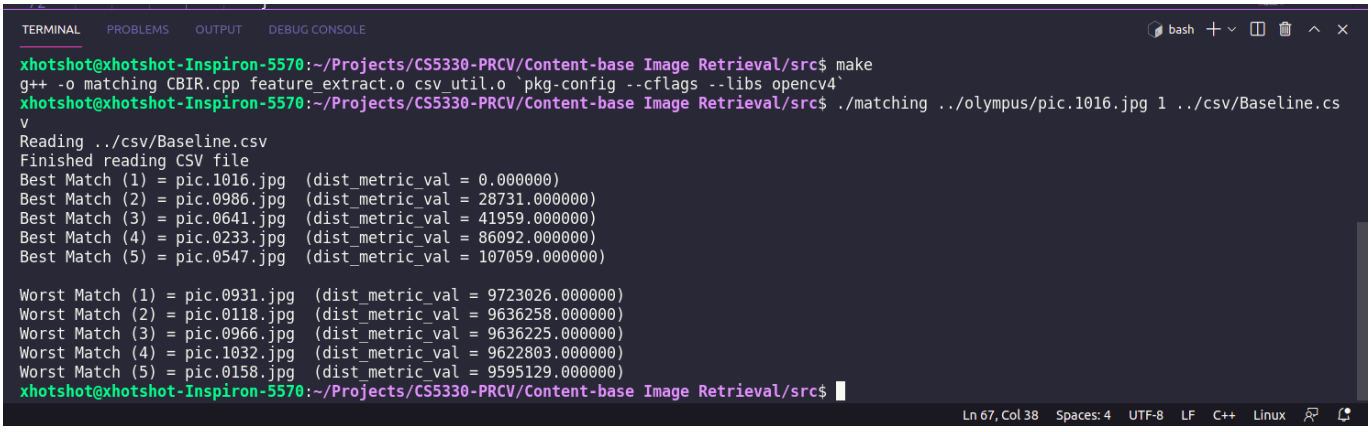
Sreehari Premkumar

Description of the Project

Project 2 of Pattern Recognition and Computer Vision is focused on the development of a Content-Based Image Retrieval (CBIR) system using OpenCV and C++. The goal of a CBIR system is to retrieve similar images to a given target image by comparing their visual content.The first step of the project would be to input a set of images into the system. These images are used to build feature vectors that represent the visual content of each image. These feature vectors can be constructed using different methods, such as directly accessing pixel values, color histograms, texture analysis using the Laws filter etc.Once the feature vectors have been constructed, they can be used to compare a target image to the images in the dataset. The system would take a target image as input and compare its feature vector to the feature vectors of all the images in the dataset. The images with feature vectors that are most similar to the target image's feature vector are returned as results.Different tasks in the project would use different features of the image as the feature vector, providing different results. By comparing the efficiency of different methods, we can gain insights into which methods are most effective for CBIR.

TASK - 1

Baseline Matching - 9x9 square in middle of image :



The above image shows Terminal output of Baseline Matching

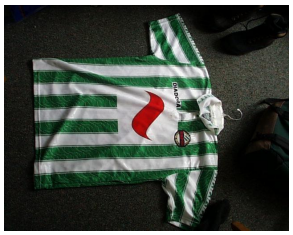
The distance Metric used here is Sum of Squared Error (SSE), and so lower the error more close the image is, hence the input image (Best Match 1) has 0.00 as it is identical

Input Image :  (Pic. 1016.jpg)

Output Images (Best Match) :

Pic. 1016

Pic.0986



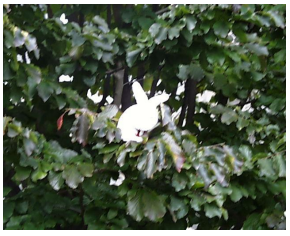
Pic.0641

Pic.0233

Pic.0547

We can see that all the images have a red color in the center of the image, which can justify the 9x9 center filter suggesting it as similar.

The following are two of the worst Match



Pic. 0931

Pic. 0118

They both have white color in the center

TASK - 2

Histogram Matching

Case 1: RG Histogram, 16 bins

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE                                  bash

xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$ ./matching ../olympus/pic.0164.jpg 2 ../csv/Hist_RG.csv


Bin size :
16
Reading ../csv/Hist_RG.csv
Finished reading CSV file
Best Match (1) = pic.0164.jpg  (dist_metric_val = 0.999061)
Best Match (2) = pic.0268.jpg  (dist_metric_val = 0.543081)
Best Match (3) = pic.0933.jpg  (dist_metric_val = 0.535550)
Best Match (4) = pic.0080.jpg  (dist_metric_val = 0.472625)
Best Match (5) = pic.0871.jpg  (dist_metric_val = 0.456291)

Worst Match (1) = pic.0015.jpg  (dist_metric_val = 0.003956)
Worst Match (2) = pic.0021.jpg  (dist_metric_val = 0.006658)
Worst Match (3) = pic.0022.jpg  (dist_metric_val = 0.007702)
Worst Match (4) = pic.0018.jpg  (dist_metric_val = 0.019122)
Worst Match (5) = pic.0503.jpg  (dist_metric_val = 0.024001)
xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$

Ln 67, Col 38  Spaces: 4  UTF-8  LF  C++  Linux
```

The above image shows Terminal output of 16 Bin, RG Histogram Matching

The distance Metric used here is Intersection of Histogram values, and so higher the value more close the image is, hence the input image (Best Match 1) has close to 1 and is same as input image

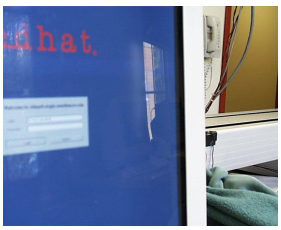
Input Image :  (Pic. 0164.jpg)

Output Images (Best Match) :

Pic. 0164

Pic.0268

Pic.0933

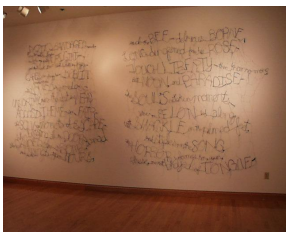


Pic.0080

Pic.0871

The Output kind of makes sense as the histogram was RG and there was no blue component, so the input image with lots of blue would be a bad input for the same, and hence it matched images of other color combinations

The following are three of the worst Match



Pic. 0015

Pic. 0021

Pic. 0022

All the worst images have a Red hue for some reason

TASK - 2

Histogram Matching

Case 2: RGB Histogram, 8 bins

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE                                  bash

xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$ ./matching ../olympus/pic.0164.jpg 2 ../csv/Hist_RGB.cs
v

Bin size :
8
Reading ../csv/Hist_RGB.csv
Finished reading CSV file
Best Match (1) = pic.0164.jpg  (dist_metric_val = 0.999010)
Best Match (2) = pic.0110.jpg  (dist_metric_val = 0.424420)
Best Match (3) = pic.1032.jpg  (dist_metric_val = 0.423683)
Best Match (4) = pic.0092.jpg  (dist_metric_val = 0.389066)
Best Match (5) = pic.0976.jpg  (dist_metric_val = 0.373282)

Worst Match (1) = pic.0015.jpg  (dist_metric_val = 0.005795)
Worst Match (2) = pic.0503.jpg  (dist_metric_val = 0.007959)
Worst Match (3) = pic.0021.jpg  (dist_metric_val = 0.009782)
Worst Match (4) = pic.0022.jpg  (dist_metric_val = 0.010395)
Worst Match (5) = pic.0048.jpg  (dist_metric_val = 0.017357)
xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$
```

The above image shows Terminal output of 8 Bins, RGB Histogram Matching

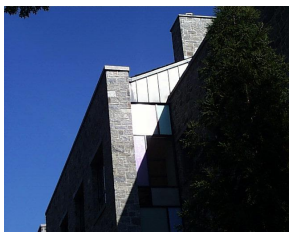
The distance Metric used here is “Intersection” of Histogram values, and so higher the value more close the image is, hence the input image (Best Match 1) has close to 1 and is same as input image

Input Image :  (Pic. 0164.jpg)

Output Images (Best Match) :

Pic. 0164

Pic.0110



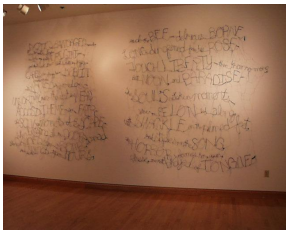
Pic.1032

Pic.0092

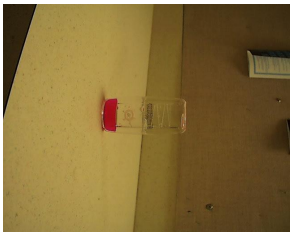
Pic.0976

All the outputs have a variation of the same blue and whitish gray in them.

The following are three of the worst Match



Pic. 0015



Pic. 0503

Pic. 0021

Worst case have a completely different color palette

TASK - 3

Multi - Histogram Matching

Color histogram the whole image was taken, and a percentage of the image from the center was taken, the percentage was decided by a parameter (value was 50% in the result shown). Then the weighted average of both of their Histogram Intersections was taken, the weight was 0.5 for both images (equally weighted).

```
xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$ ./matching ../olympus/pic.0274.jpg ../csv/MultiHist.csv

usage: ./matching <image path> <feature_set_number> <feature_set_file_csv>
xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$ ./matching ../olympus/pic.0274.jpg 3 ../csv/MultiHist.c
sv
Reading ../csv/MultiHist.csv
Finished reading CSV file
Best Match (1) = pic.0274.jpg  (dist_metric_val = 0.998763)
Best Match (2) = pic.0273.jpg  (dist_metric_val = 0.629854)
Best Match (3) = pic.0209.jpg  (dist_metric_val = 0.625160)
Best Match (4) = pic.0881.jpg  (dist_metric_val = 0.623264)
Best Match (5) = pic.1055.jpg  (dist_metric_val = 0.613465)

Worst Match (1) = pic.0015.jpg  (dist_metric_val = 0.005088)
Worst Match (2) = pic.0021.jpg  (dist_metric_val = 0.009257)
Worst Match (3) = pic.0022.jpg  (dist_metric_val = 0.009526)
Worst Match (4) = pic.0025.jpg  (dist_metric_val = 0.021993)
Worst Match (5) = pic.0503.jpg  (dist_metric_val = 0.028370)
xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$
```

The above image shows Terminal output of 8 bit Multi-histogram

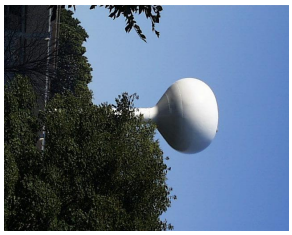
The distance Metric used for both versions of the image is Intersection of Histogram, and so higher the value the closer the image is, hence the input image (Best Match 1) has close to 1 and is the same as input image.

Input Image :  (Pic. 0274.jpg)

Output Images (Best Match) :

Pic. 0274

Pic.0273



Pic.0209

Pic.0881

Pic.1055

The worst match was similar to the previous output.

This system gives (adjustable) additional weight to the center of the image along with the whole image. We can see from the results that the system is quite accurate.

If at some point we need to focus more on the center we can adjust the parameter:

Weight_distance in the code to less than 0.5, so it will reduce the weight for the whole image and give more focus to the center part.

If we need to adjust the size of the center block of the image taken, we can adjust that by changing the center_square_break parameter, which decides how much percentage size of the original image should be taken.

0.5 gives half the size of the image from the center, 1.0 gives the whole image itself.

TASK - 4

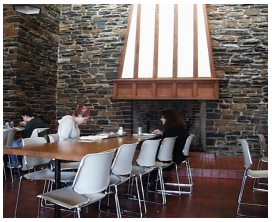
Texture and Color Histogram

Color histogram the whole image was taken, and for texture feature, magnitude of sobel-X and sobel-Y was calculated. Then Intersection was used as distance metric for the color histogram, and Sum of squared errors for the magnitude image. The weighted average of the distance metric was taken, the weight for the distance metric was 0.35, stating that 0.35 weight is for the color histogram and 0.65 is for the magnitude image.

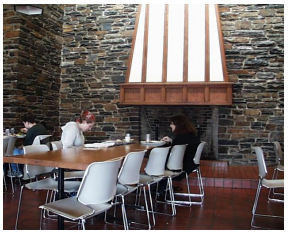
```
109             if (sorted[j].sum <= sum)
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE                                  bash
xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$ ./matching ../olympus/pic.0535.jpg 4 ../csv/TextureColo
rHist.csv
Reading ../csv/TextureColorHist.csv
Finished reading CSV file
Best Match (1) = pic.0535.jpg  (dist_metric_val = 0.349437)
Best Match (2) = pic.0628.jpg  (dist_metric_val = 0.251622)
Best Match (3) = pic.0171.jpg  (dist_metric_val = 0.250711)
Best Match (4) = pic.0605.jpg  (dist_metric_val = 0.245725)
Best Match (5) = pic.0829.jpg  (dist_metric_val = 0.244772)

Worst Match (1) = pic.0251.jpg  (dist_metric_val = -0.108643)
Worst Match (2) = pic.0823.jpg  (dist_metric_val = -0.090713)
Worst Match (3) = pic.1009.jpg  (dist_metric_val = -0.085865)
Worst Match (4) = pic.0511.jpg  (dist_metric_val = -0.083325)
Worst Match (5) = pic.0250.jpg  (dist_metric_val = -0.052003)
Ln 227, Col 65  Spaces: 4  UTF-8  LF  C++  Linux
```

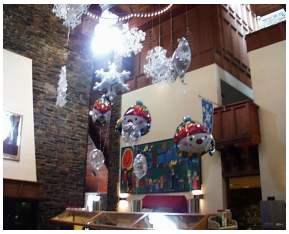
The above image shows Terminal output of Texture and Color Histogram

Input Image :  (Pic. 0535.jpg)

Output Images (Best Match) :



Pic. 0535



Pic.0628

Pic.0171

Pic.0605

Pic.0829

The following are three of the worst Match

Pic. 0251

Pic. 0823

Pic. 1009

Both the distance metric was normalized and while one was added the other one was subtracted as Intersection needs to be high value and SSE needs to be low for the images to match.

We can see from the output that there is importance given to the similarity in patterns, although the matching might not be as perfect in terms of texture, due to the fact that magnitude and sum of squared error was used, but it is pretty fine.

TASK - 5 + Extension (Laws Filter)

Custom Design

A new dataset was taken containing Landscape, the link to the dataset:

https://www.kaggle.com/datasets/arnaud58/landscape-pictures

The dataset had 4500+ images of various sizes and quality, which was a lot to be uploaded on github, so only the first 1327 images were taken, of which around 1300 were used for training and 27 for testing.

For Texture Analysis, Laws Filter was used.

L5 = {1, 4, 6, 4, 1}

E5 = {1, 2, 0, -2, -1}

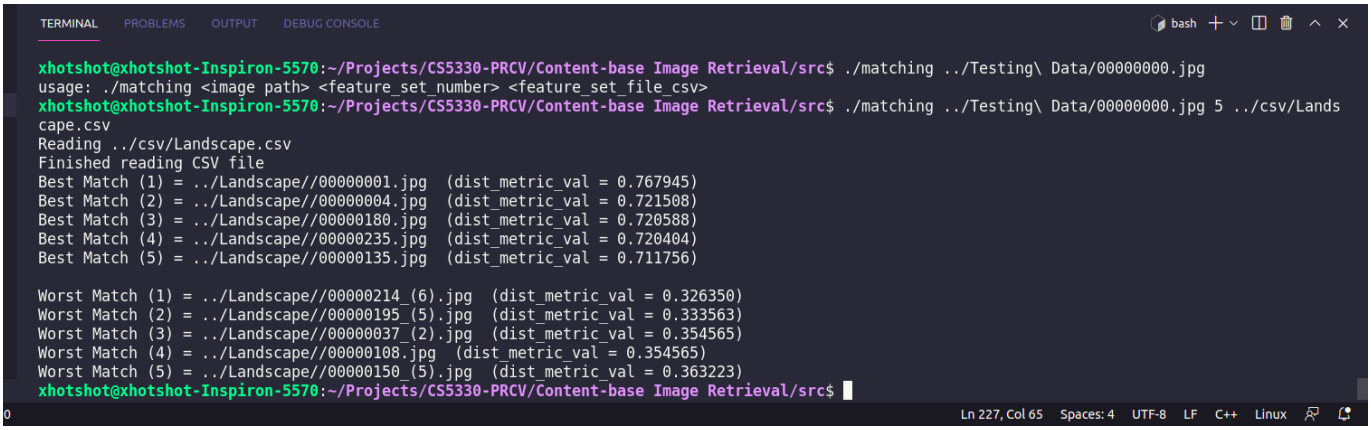
S5 = {-1, 0, 2, 0, -1}

W5 = {1, -2, 0, 2, -1}

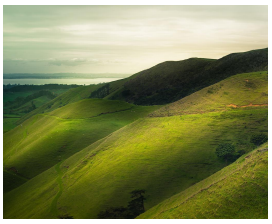
R5 = {1, -4, 6, -4, 1}

25 combinations of the above-mentioned linear filters were used and 8 bin histograms of each were taken, to make a feature vector of size 200.

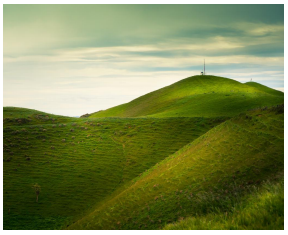
Test Image 1



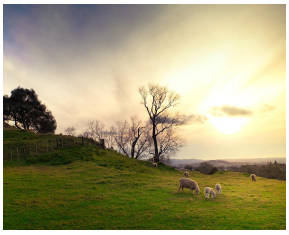
The above image shows Terminal output of Laws Filter + Color Histogram

Input Image :  (Pic. 00000000.jpg)

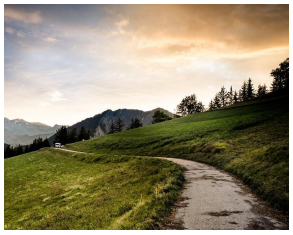
Output Images (Best Match) :



Pic.00000001



Pic.00000004

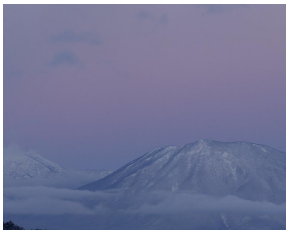


Pic.00000180

Pic.00000235

Pic.00000135

The following are three of the worst Match



Pic.00000214_(6)

Pic.00000195_(5)

Pic. 00000037_(2)

The distance metric used for both the texture and color was histogram intersection.

Also a weighted average of both of them were taken with the weights being 0.5 each(equally weighted) for the results above.

Test Image 2

```
325             }
326             else

TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE                                  bash

xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$ ./matching ../Testing\ Data/00000010_\(2\).jpg 5 ../csv
/Landscape.csv
Reading ../csv/Landscape.csv
Finished reading CSV file
Best Match (1) = ../Landscape//00000120_(2).jpg  (dist_metric_val = 0.997989)
Best Match (2) = ../Landscape//00000188_(5).jpg  (dist_metric_val = 0.747593)
Best Match (3) = ../Landscape//00000008_(3).jpg  (dist_metric_val = 0.728528)
Best Match (4) = ../Landscape//00000108_(2).jpg  (dist_metric_val = 0.716771)
Best Match (5) = ../Landscape//00000136_(3).jpg  (dist_metric_val = 0.716173)

Worst Match (1) = ../Landscape//00000214_(6).jpg  (dist_metric_val = 0.275364)
Worst Match (2) = ../Landscape//00000195_(5).jpg  (dist_metric_val = 0.287582)
Worst Match (3) = ../Landscape//00000017_(5).jpg  (dist_metric_val = 0.302974)
Worst Match (4) = ../Landscape//00000058_(4).jpg  (dist_metric_val = 0.316464)
Worst Match (5) = ../Landscape//00000132_(3).jpg  (dist_metric_val = 0.333429)
xhotshot@xhotshot-Inspiron-5570:~/Projects/CS5330-PRCV/Content-base Image Retrieval/src$

0                                  Ln 323, Col 14  Spaces: 4  UTF-8  LF  C++  Linux
```

The above image shows Terminal output of Laws Filter + Color Histogram

Input Image :  (Pic. 00000010_2.jpg)

Output Images (Best Match) :

Pic.00000120_2

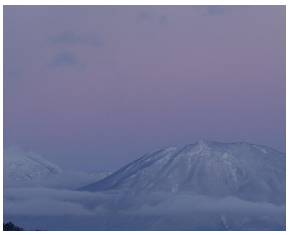
Pic.00000188_5

Pic.00000008_3

Pic.00000108_2

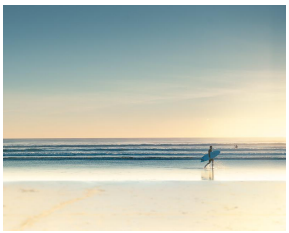
Pic.00000136_3

The following are three of the worst Match



Pic.00000214_(6)

Pic.00000195_(5)



Pic. 00000017_(5)

The first best image is identical.. The dataset had repeated images, which I got to know after the search.

Other than that the images are pretty close and the texture effect is very close.

Reflection

The project started off easy, but became more challenging as I progressed. I appreciated the difficulty because it helped me learn more about c++ and opencv, which I had never used before. I was impressed with how effective the color histogram was. I struggled with the csv file and distance metric, as well as incorporating my code from a previous project, but managed to overcome those challenges. The use of sobel magnitude for texture was a great idea, and the implementation of laws filter for texture produced surprisingly good results. Overall, I enjoyed the project and gained a lot of knowledge from it.

Acknowledgement

The dataset and csv reading code provided by the Instructor was useful.

The Landscape dataset from kaggle (Link is above)

Referenced Stack overflow and geeksforgeeks to help me with debugging the code.

Referred OpenCV official website to write the code.