CS365CV Project 2 " Content-Based Image Retrieval" by Mike Z. and Heidi H.

Project Structure:

cbirgui (main):

recursively read files from a directory.

display GUI

process query image

process database image

show result for top 20 matches and update.

(comparator for comparing similarity.)

Img class:

each image class object contains:

an image class has a path to the original image,

a flag for checked/unchecked (for future use),

a similarity value that shows its similarity to the query image. The larger the value is, the better the match it is.

When comparing pictures, call related histogram function from the histogram library. And calculate the distance metric within the image class. Return the similarity value to the main function.

Histogram library:

the class that processes all histogram functions. the functions usually take in a given path to an image and return a histogram.

Compilation and Running

Makefile - GUI : make cbirgui

Run: ../bin/cbirgui ../data/<queryImageName> <database>

Makefile - CommandLine: make cbir

Run: ../bin/cbir ../data/<queryImageName> <database> <# of image to report> <method index>

GUI:

the trackbar on top of the images enables a user to select matching methods, according to the following rules:

0- baseline matching

1- baseline hue-saturation histogram matching

2- multiple histograms matching

3- Sobel filtering + color matching

4- color matching utilizing earth mover distance metric (expensive, takes 20 mins & to be improved)

5- Laws filter subset + color matching

6- Fourier transform + color matching (logically unsound)

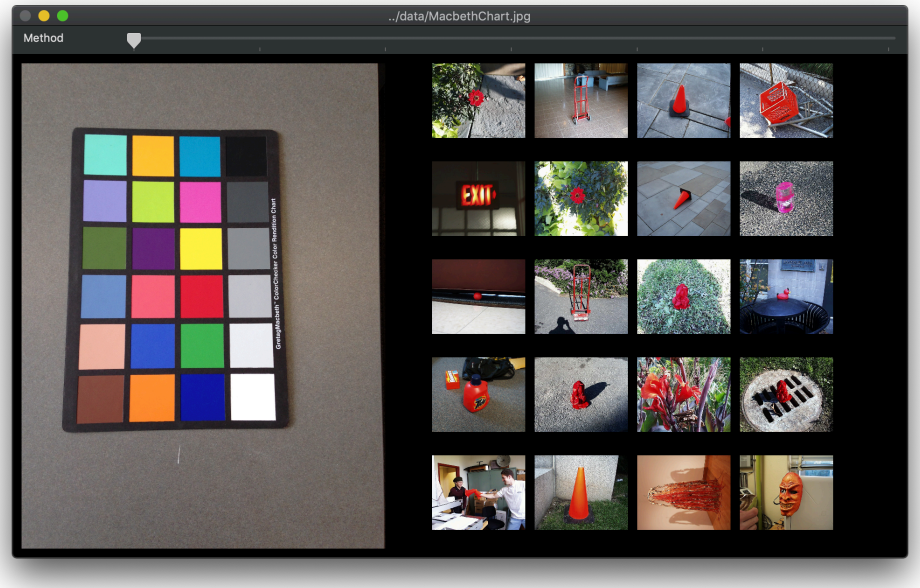
7- rgb + saturation in weighted average

Tasks:

Task1: Baseline Matching

The baseline matching method compares a 5x5 block of pixels from the center of the two images and finds the SSD. The smaller the SSD is, the more similar the two pictures are.

This is our result for this method:

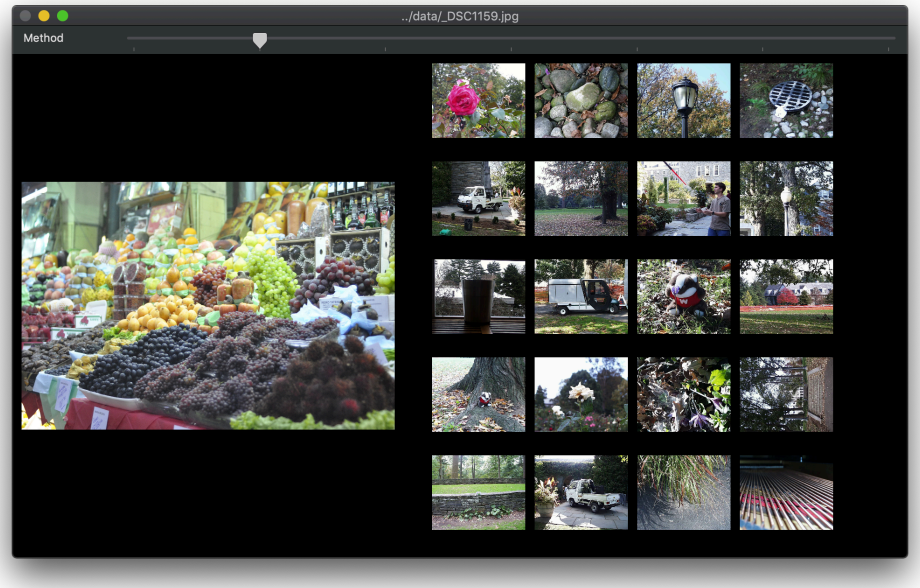


As shown, this method only considers the 5*5 block in the center. In this case, the center of the Macbeth chart picture is red, and all the top matches have red centers.

Task2: Baseline Histogram Matching

Out baseline-histogram method takes compares the whole hue-saturation histogram of two images. We use INTERSECT distance metric.

This is our result:

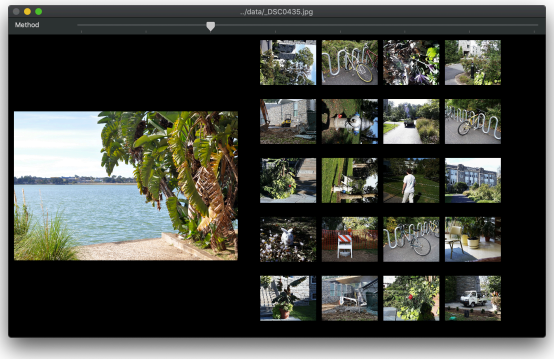


As shown, the matches are similar in that it has a green-dark color scheme, just as the query image.

Task3: Multiple Histogram Matching

For the multiple histogram matching task, we create multi whole hue-saturation histogram on edges and the center. (shown in the following illustration) The logic behind this is that the center and the four corners of an image sometimes occupy more visual weight.

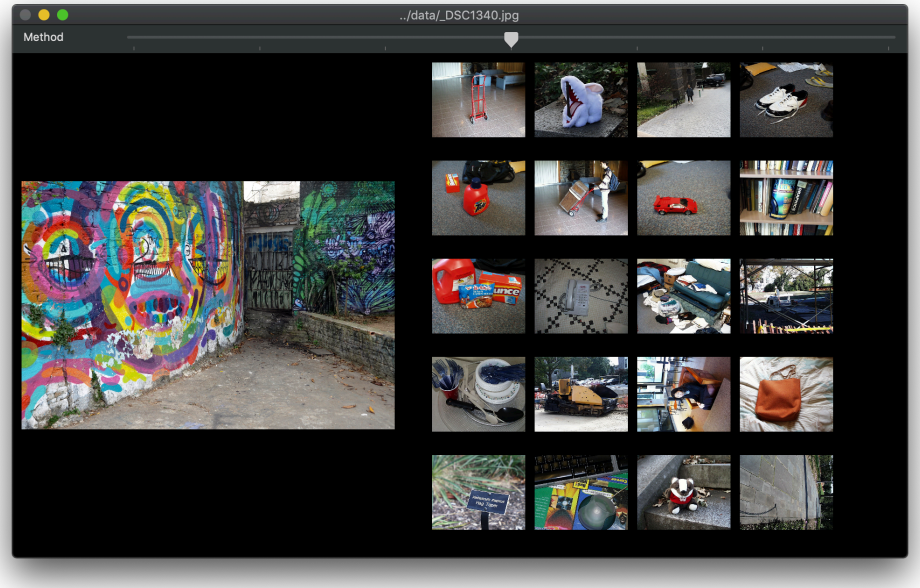
The result:



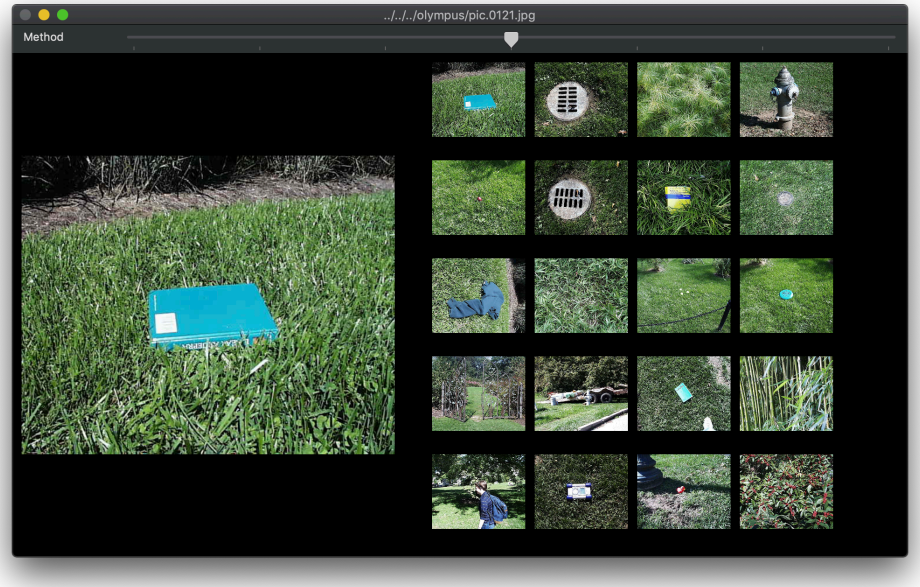
The matching results show evidence of green around the edges but the unsaturated blue over the center is not very obvious within most of the results.

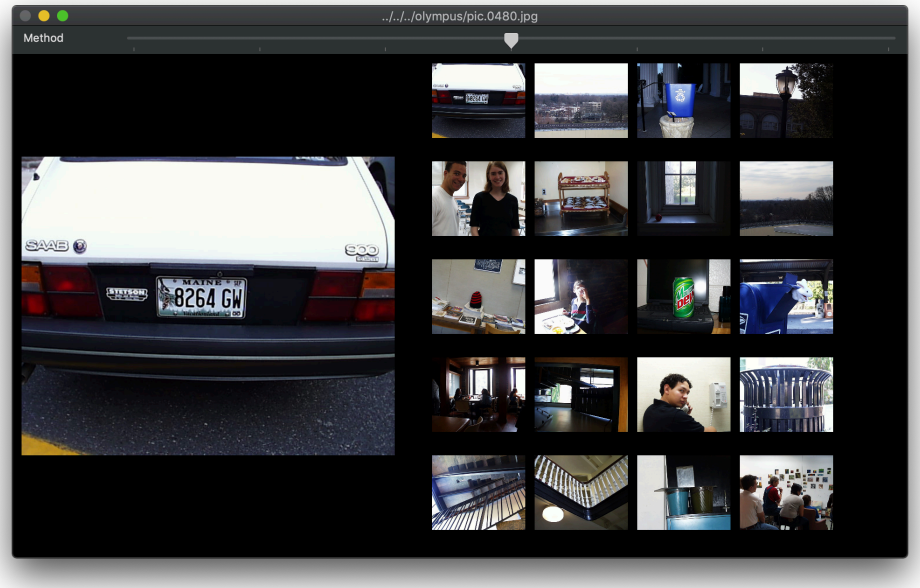
Task4: Color and Texture

In this project, we applied the Sobel filters for texture information. And using the hue-saturation histogram for color information. We weighted the average of both information. This is the result:



The result does show some similar texture features, but we cannot tell how well it works. We therefore used two other tests, which will also be used to test our extensions.



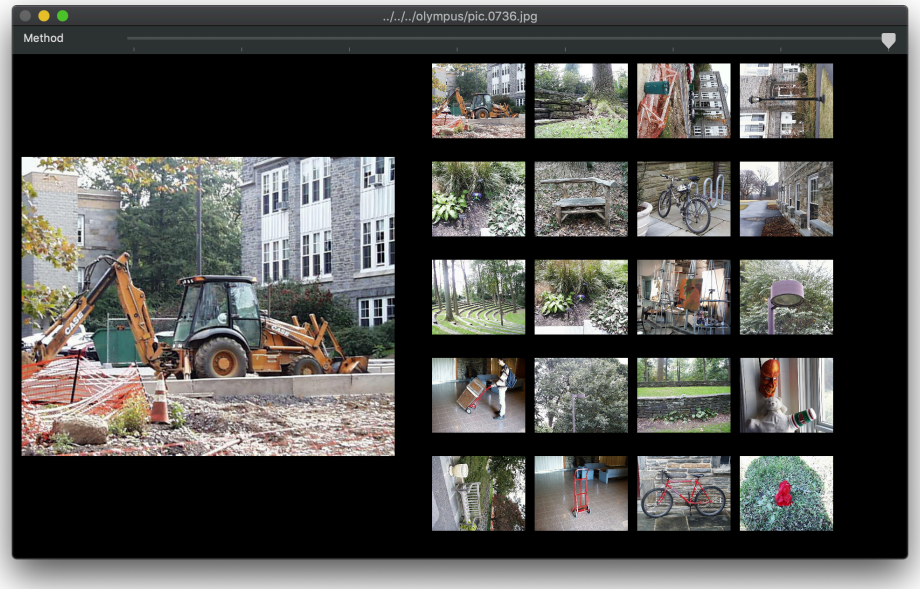


The matches of the first picture all have grass in it, which has both simple color property and texture property. This Sobel+HS method performs well. In a more difficult task, where the query image is a license plate, it does not do well, as it did not find other license plates in the database.

Task5: designed distance Metric

R2GB + saturation. This is a self-designed color matching method that considers the red, green, blue channel while emphasizing the overall saturation of the image.

Since green takes more visual weight from our eyes' sensor, we gives the green channel twice the visual weight than red and blue.



The results evident an overall matching saturation. Since green takes more visual weights, the green channel in the results seems to demonstrate similar value and saturation.

Extensions:

Extension1: more efficient file reader and parser.

We use a recursive function to read image files from the database folder. The function first read 16 images and if there is still more images to read, it doubles the number each recursion until finish reading all files within the destinate folder.

Extension2: self-designed texture and color metric involving Laws Texture Energy Measures

Info page about Law's texture (https://en.wikipedia.org/wiki/Image_texture#Laws_Texture_Energy_Measures)

Laws Texture Energy Measures [edit]

Another approach is to use local masks to detect various types of texture features. Laws[4] originally used four vectors representing texture features to create sixteen 2D masks from the outer products of the pairs of vectors. The four vectors and relevant features were as follows:

```
L5  =  [ +1  +4  6  +4  +1 ]  (Level)
E5  =  [ -1  -2  0  +2  +1 ]  (Edge)
S5  =  [ -1   0  2   0  -1 ]  (Spot)
R5  =  [ +1  -4  6  -4  +1 ]  (Ripple)
```

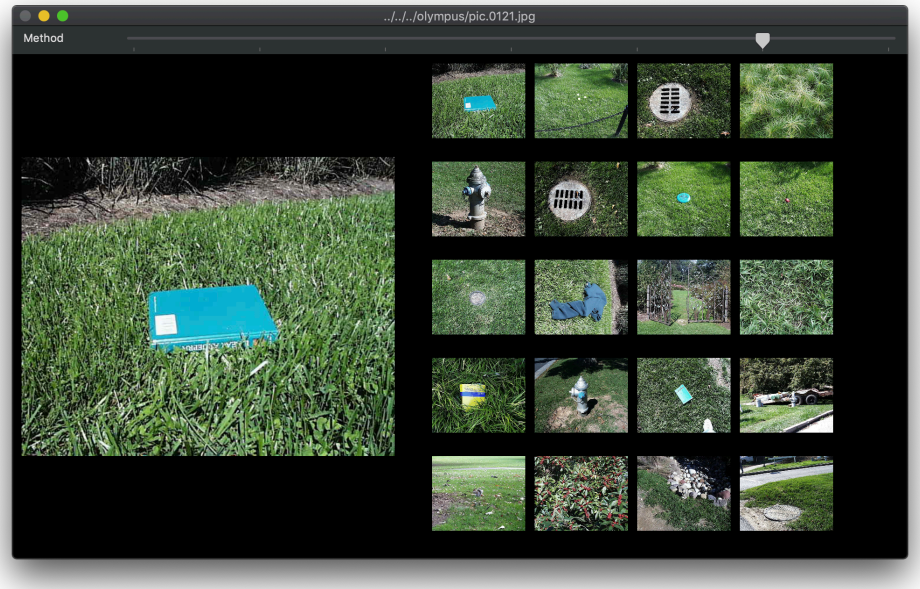
To these 4, a fifth is sometimes added:[5]

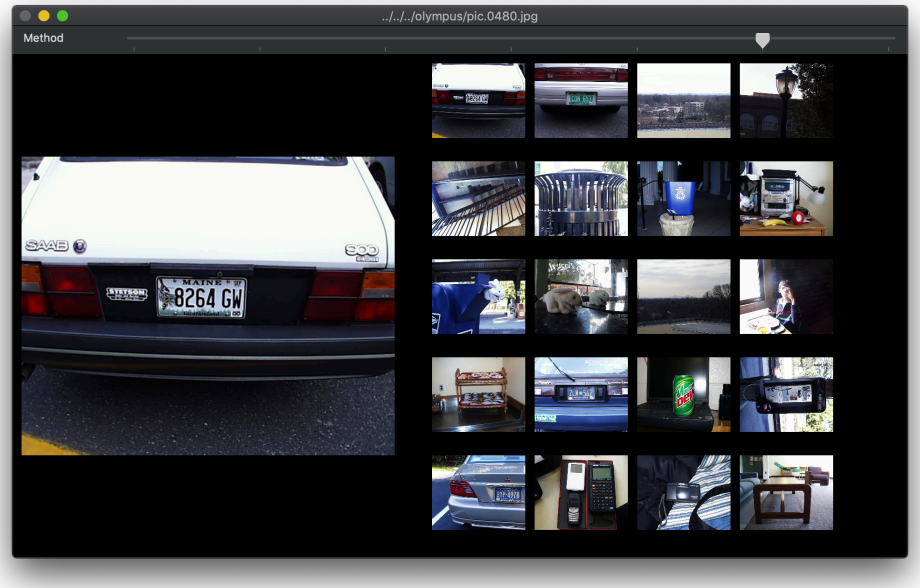
```
W5  =  [ -1  +2  0  -2  +1 ]  (Wave)
```

From Laws' 4 vectors, 16 5x5 "energy maps" are then filtered down to 9 in order to remove certain symmetric pairs. For instance, L5E5 measures vertical edge content and E5L5 measures horizontal edge content. The average of these two measures is the "edginess" of the content. The resulting 9 maps used by Laws are as follows:[6]

```
L5E5/E5L5
L5R5/R5L5
E5S5/S5E5
S5S5
R5R5
L5S5/S5L5
E5E5
E5R5/R5E5
S5R5/R5S5
```

We applied e5l5 for horizontal edges, l5e5 for vertical edges, l5r5, r5l5, and s5s5 individually to the original image and computes five histograms. We then used those histograms in our comparison.

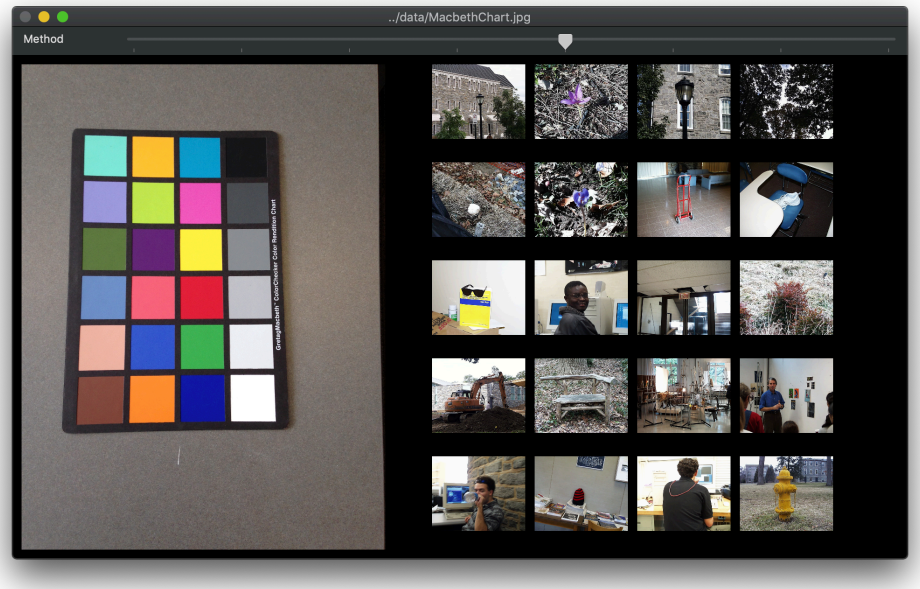




Again, as the grass query image is fairly simple, the method performs fairly well. The method also behaves surprisingly well in the license plate task, in which it found 3 more license plates.

Extension3: Earth Mover's distance metric

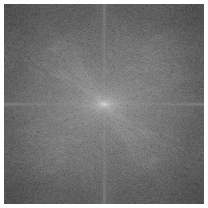
We tried earth mover's distance metric for comparing the hue-saturation histogram. It is an expensive method that took a lot of computational power especially for processing a database with 1000+ images.



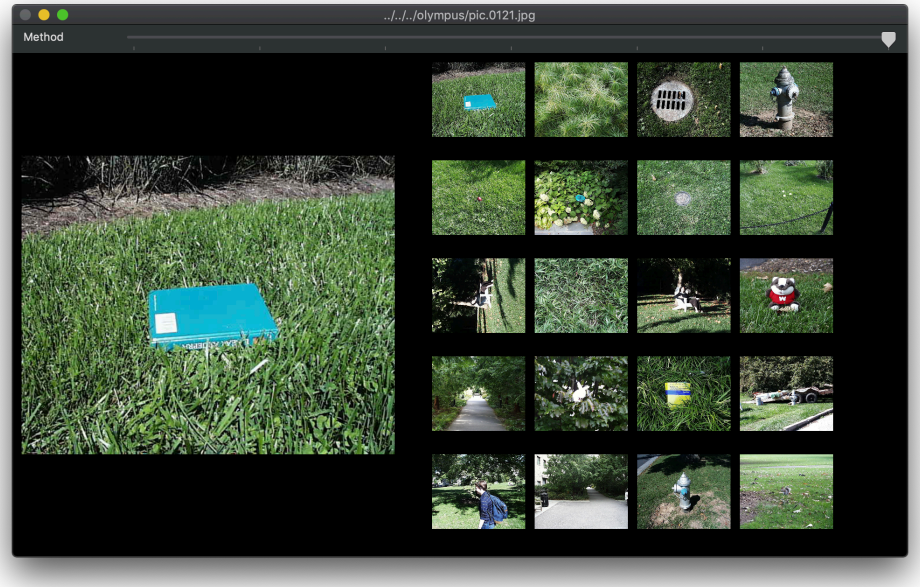
The method is extremely costly. It could be further improved and tested in addition to hue-saturation histograms.

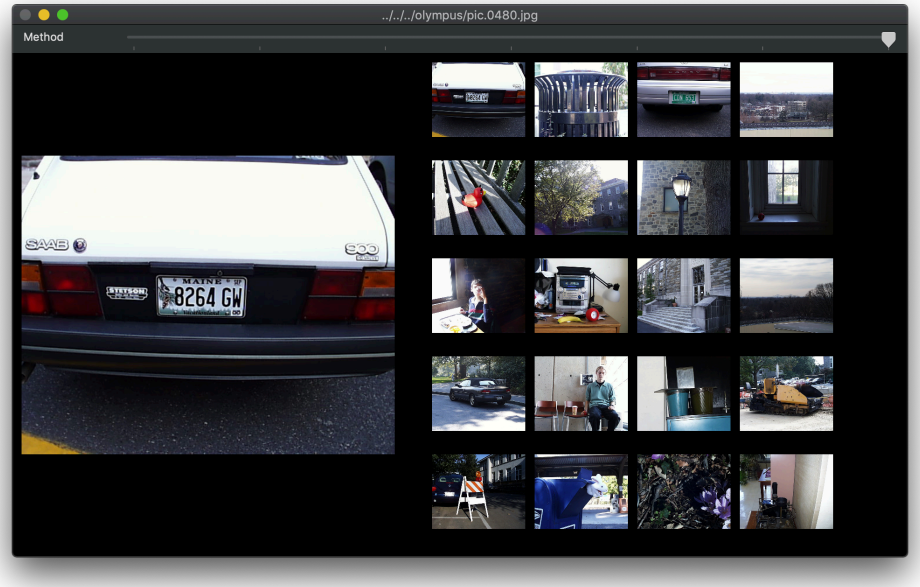
Extension4: Fourier transform

We applied Fourier transform to the original image and computed histograms of the output to use in comparison. This method also takes into account of color (Hue-saturation)

In retrospect, we think that this is not an effective metric. The spatial distribution of the pixels of the Fourier-transformed image carries the information of the texture. By calculating histograms, we do not really make use of that spatial information. However, comparing Fourier-transformed images directly is also difficult, because different images will have different sizes. One possible way around this will be to directly compare Fourier transform of small sections of two images.





This method worked fine, but we suspect color histogram similarity is the driving force.

Extension5: GUI

We introduce a trackbar that enables the user to select which matching method to use. The interface also displays the best 20 matches in grids, to the right of the query image.

Acknowledgment:

This is a group project by Mike Zheng and Heidi He.

References & Useful Links:

histogram-related functions in openCV:

https://docs.opencv.org/2.4/modules/imgproc/doc/histograms.html?highlight=calchist

texture filters:

https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect12.pdf