Project 2 : Content-based Image Retrieval

The purpose of the project is to understand different ways of finding similarities between images based on color, texture, spatial, etc, and create a pipeline, where given an image and count of numbers it returns the top N similar images based on the method of comparison. This project helped us to understand how to create our own feature extractor based on the problem at hand, for example, to find a banana or green bin we can give more weight to green color and use that knowledge to create our own feature.

I am creating my pipeline such that given an image and a particular task, I take input from the user the CSV file to be used, if the CSV file is not provided then I search if the CSV file is present in the image directory, if yes it uses that particular feature CSV. If the CSV is not present then it calls the feature function and writes the features into a new CSV. And when the search is done for the next image it uses the already created CSV instead of making the CSV from the scratch for that particular task. The Task buttons won't work unless the image and the image folder are selected. The default value for the top N is 3, it can be changed on the screen, at most 7 images will be shown as output, but based on the N, the name of all the top N similar images will be printed.

I am using a modified bubble sort for fetching the top N images. The bubble sort runs in O(# of images).

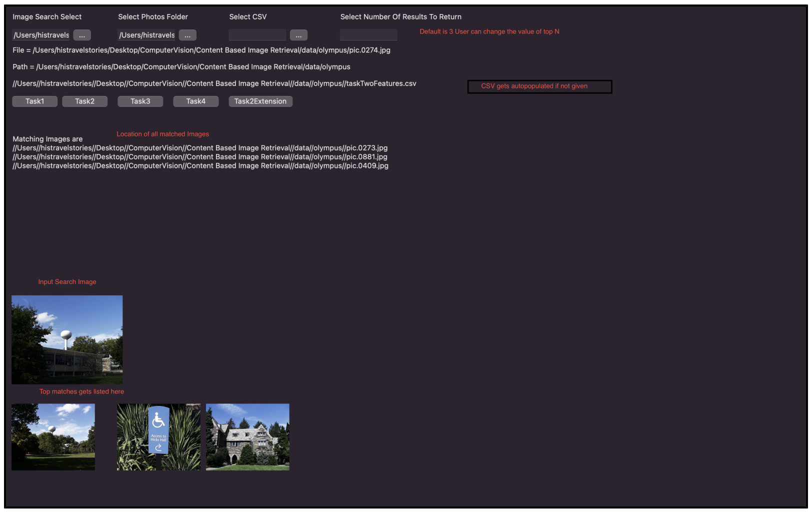
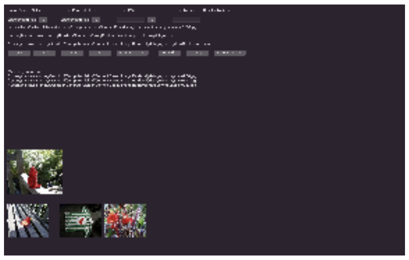
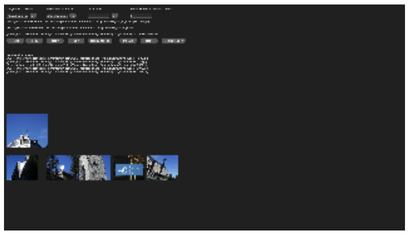
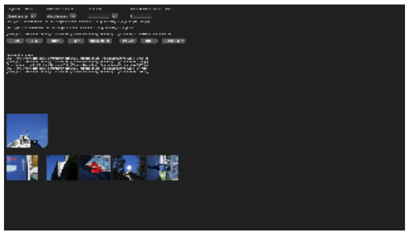
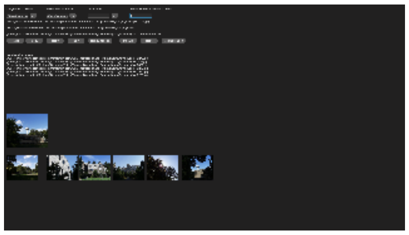
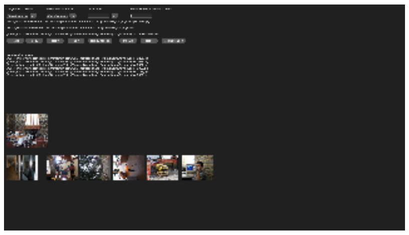
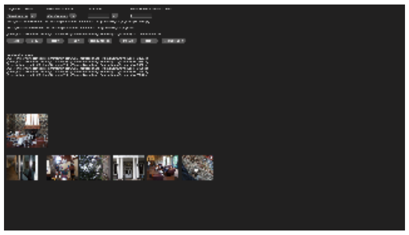
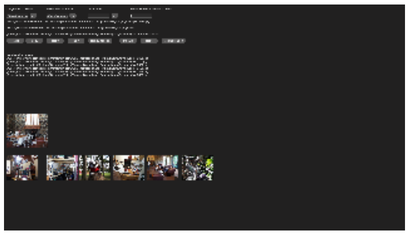
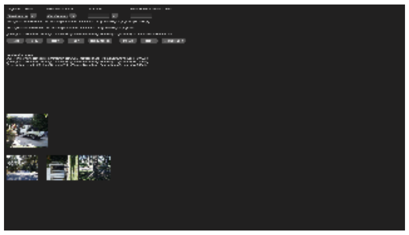
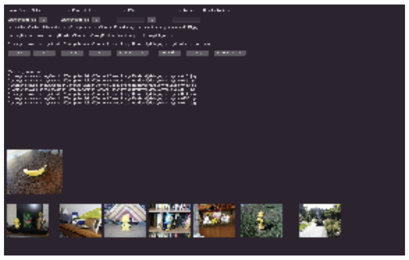
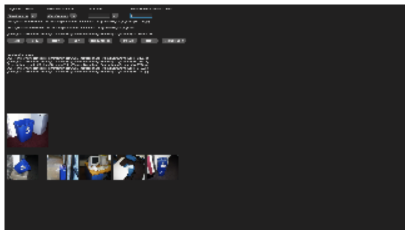


Image Search Select
Select Photos Folder
Select CSV
Select Number Of Results To Return
/Users/histravels
...
/Users/histravels
...
...
Default is 3 User can change the value of top N
CSV gets autopopulated if not given
Task1
Task2
Task3
Task4
Task2Extension
Location of all matched Images
Input Search Image
Top matches gets listed here
Access to Hicks Hall

```
File = /Users/histravelstories/Desktop/ComputerVision/Content Based Image Retrieval/data/olympus/pic.0274.jpg
Path = /Users/histravelstories/Desktop/ComputerVision/Content Based Image Retrieval/data/olympus
//Users//histravelstories//Desktop//ComputerVision//Content Based Image Retrieval//data//olympus//taskTwoFeatures.csv
Matching Images are
//Users//histravelstories//Desktop//ComputerVision//Content Based Image Retrieval//data//olympus//pic.0273.jpg
//Users//histravelstories//Desktop//ComputerVision//Content Based Image Retrieval//data//olympus//pic.0881.jpg
//Users//histravelstories//Desktop//ComputerVision//Content Based Image Retrieval//data//olympus//pic.0409.jpg
```

Task	Button	Comment	Result
Baseline Matching	Task 1	Here the central 9*9 pixels are selected as features and then the sum of the square is calculated with the source image and the rest. The least N values are the N most similar images.	
Histogram Matching 3D	Task 2	A 3D normalized color histogram is created as a feature for each vector. Histogram intersection is used as a distance-vector to identify similar images. 1 is subtracted from the value so that the similar one is the one with a minimum value. (The result is swapped from the example given in the question because the values differ in 0.002 between the two images). Showing the top N input taken from the user here.	
Histogram Matching 2D	Extension	A 2D histogram is created with rg chromaticity using 16 bins, while dividing I have added 1 to remove corner-case errors.	

Multi-Histogram Matching	Task 3	I have divided the image into 2 spatial parts, the top half and the bottom half. Both are given equal weights and on the resultant feature vector, the histogram intersection distance metric is used. The GUI removes and adds results on the screen based on the count given in top N.	
Texture and Color (Sobel Magnitude)	Task 4	Similar to Task 3 instead of spatial vectors, we are using a 3D normalized color histogram and Sobel magnitude. The difference we see here is that the texture or the images are taken into consideration due to the introduction of the Sobel magnitude. And we see more similar images than the task 2 and task 3 methods of search.	Task 2 Results for 0535.jpg  Task 3 Results for 0535.jpg  Task 4 Results for 0535.jpg 
Texture and Color (Laws Filter)	Task 4 Extension	We used the Laws filter instead of the Sobel magnitude. The E5L5 (Gaussian Level5 = [1 4 6 4 1], Gradient Edge = [-1 -2 0 2 1], is convolved and the kernel is used in the image to get the processed image, on which color 3D histogram is applied.	
Find All Banana Images	Bananas Extension	For Finding Bananas Figures, we are taking a portion of the image cropping it with (200,200) coordinate and with width=200 and height =200. Then find the yellow portion of the image. I figured out that this might not be the best method as there might be other objects with yellow color. Added gradient magnitude along with this but could not see much improvement on the result. Maybe reducing the size of the image again to a specific portion would improve the detection. Could find one banana image out of 4, the ducks look like bananas so if that is included then I have better results. Checked the results through the SumOfAbsoluteDifference as well.	

Find Blue Bin Images	Task 5, Blue Bin Extension	Here the image is converted to HSV format to identify the blue sections of the image. inRange() function is called to identify the binary format of the blue shades values in the image. I convert the image back to the RGB and apply erode effect with MORPH_RECT with a size of (40,40). The resultant image is masked with the initial image. After which the rg chromaticity is calculated for the resultant masked image where only the blue sections are visible. This is used as a feature vector along with the histogram intersection to identify similar images. Se The resultant images fetched are almost a good response as it fetches the blue tables that look like a bin wrt to dimensionality.	
GUI	Extension	GUI is developed using wxWidgets	

Learning Outcome

At the completion of Project 2, the following are my learning outcomes:

- Learned to develop a makeshift GUI using wxwidgets (beginner) to make the output look better than a command line.
- Understood the thinking process of how to select features given the images to search from beforehand.
- Learned how to develop the pipeline for comparing the feature vectors for the source image and the database image and finding different distance metrics like the sum of the square, histogram intersection, etc.
- Learned how the histograms can be constructed in 2D (rg chromaticity) and 3D(RGB) format.
- Used texture filters to improve the search process and learned how to use multiple histograms with different weights to create features.

Acknowledgment

- wxgui : Lot of resources, StackOverflow but the best lib was: https://github.com/gammasoft71/Examples_wxWidgets/blob/master/src/CommonControls/Button/Button.cpp
- Stalking the Piazza at times to get some ideas of what I am missing or help me understand questions better.
- Understanding the texture filters: https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect12.pdf

Reflection

The image pipeline created for the previous project helped me in checking out the filters that I had in mind and visualizing how it would look, before adding it to the current project and comparing the results. Had to take a longer route and spend most of the time implementing the GUI as Mac M1 chip support tutorials are none. But once the GUI was set it was easy for me to implement lots of stuff, I see that lot of redundant codes are present in the GUI code and the feature code, but in order to separate the codes for testing each task randomly I have kept it in that manner. In the future, the code can be modularized. Sometimes its just a hit and miss, trying various methodologies to find which will suit the best for a given problem.