EECS 4422 - Computer Vision
Supriyo Shishir Ghosh – 215318728
Project - Engineering Stream

Final Report - Simple Image Search Engine

Introduction:

With the vast popularity of embedded camera devices and the continuous development of the internet technology, a noticeable growth in web sharing and photo browsing has been witnessed in the last decade. Hence came the emergence of a great number of applications based on image search. Traditionally image search can rely on text search techniques as search engines index multimedia data based on its surrounding metadata information around a photo on the web such as titles and tags. Such indexing can be highly inefficient since text words can be inconsistent with the visual content. Attention is hence drawn to content-based image retrieval (CBIR) which has achieved a great advance in the last few years. CBIR is a simple application of computer vision to the problem of search of visual content in large databases.

Existing approaches:

Three main issues when developing a CBIR algorithm must be addressed:

- Image representation
- Image organization
- And similarity measurement between images

Image representation will mainly depend on the platform used by the developer of the algorithm. It has however to satisfy the condition of being both descriptive and discriminative as it is expected to be as the algorithm will be only as efficient as its ability to distinguish similar and dissimilar images. The large visual database presents the organization as a nontrivial but rather laborious task to address. Fortunately, CBIR has information retrieval successful approaches to look up to as several CBIR algorithms rely on the commonly used inverted file structure to index large scale visual database allowing scalable retrieval. Similarity measurement between images is also critical in developing a solid code. It should reflect the relevance in semantics which is highly contradicted by the semantic. And though highly relying on both image representation and organization, image similarity measurement is based on the visual matching results with the aid of weighing schemes.

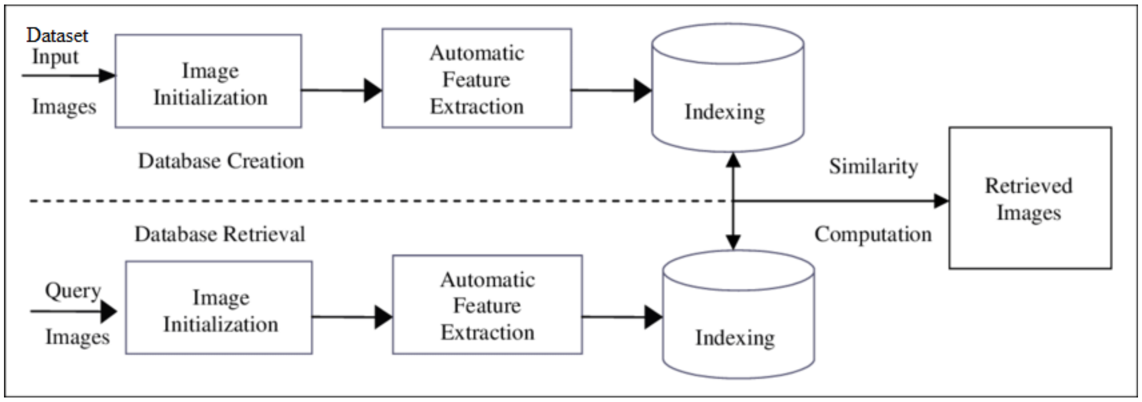


Figure: The general framework of content-based image retrieval (CBIR)

In this project, a CBIR algorithm will be developed using python as a platform where the program's input will be a query image taken from the user to retrieve similar to the given photo as an output. The image dataset has a number of 16 photos. The similarity measurement criteria can widely vary, in the project at hand however, the features that will be extracted from the image will be color based where the similarity will be based on the histogram similarity. The developed code will be discussed in detail along with the testing and evaluation of the results along with the comparison of the work with a previously designed version.

Original design (the link to the coded article is cited on the references)**:**

The original version of the design I have worked on for this project was released on December 1, 2014 by Adrian Rosenbrock. The program was written using Python version 2.x and is not executable with the latest Python version. Python packages numpy, cv2, imutils, argparse, glob, csv are used in the program where applicable.

The program consists of an image descriptor that quantifies the image by extracting the features. The image descriptor uses an HSV colour space histogram with 8 bins and the output is a feature vector. The "describe" method of the histogram returns a list of floating-point values that are used to represent and quantify the image. The "_init_" method takes a single argument.

For extracting the image features, the histogram image descriptor is applied to the dataset images. The features collected from the dataset images are then indexed in a comma-separated values (CSV) file. Two argument parsers are used to define the directory location of the dataset and the output of the indexed features (csv file).

A class that compares two feature vectors using the chi-squared distance metric is used. The distance matrix determines the similarity by checking the distance between the two feature vectors (the query image and the indexed image features) and ranks the images by sorting according to the input query image. The "search" method takes two parameters, the "queryFeatures" extracted from the query image and "limit" which is the maximum number of results to return.

Finally, for performing the search, a new argument parser for the query image is introduced. The features from the query image are extracted and then the distance matrix method is applied to find the similarity between the query image and the indexed images features). The results are displayed in a new window. This design needs the query image to be present in the dataset of images and also have its features saved in the csv file for comparison.

New design:

The new design is written using python version 3.7 and packages numpy, cv2, argparse, glob, **pickle** are used in the program where applicable. Several lines of code have been changed from the previous design to support the current version.

An RGB color space histogram is used instead of HSV with 8 bins per red, green, and blue channel. The image descriptor collects how many pixels have a Red value that falls into bin #1, a Green value that falls into bin #2 and a Blue value falls into bin #1. This process will be repeated for each combination of bins and it will be done in a computationally efficient manner. A method "describe" normalizes the histogram so that images with the same content will have the same histogram and returns the 3D histogram as a flattened array.

In this project, I have used pickle instead of csv to index the image features. Python version 3.7 has a standard library called "pickle". Pickle is a serialized way of storing python object as the exact representation of the data-frame to the disc. If we simply save a file as a csv, we are just storing it as a comma separated list. The "pickle" module keeps track of the objects it has already serialized, so that later references to the same object will not be serialized again. In pickle, data save time (pickling) and load time (unpickling) is faster than csv [4].

For finding the similarity between two feature vectors, the Euclidean distance metric has been used instead of Chi-squared distance metric. Euclidean distance is the most well known and most used distance metric. The number of correct matches is higher, and the program execution time is lower compared to Chi-squared distance [5]. Cosine distance similarity could have been used but is not a proper distance metric because it violates both the triangle inequality and the coincidence axiom. Chi-squared function is commented out on the code so that it can be used to find the difference (probably for a larger dataset).

The program for searching with the query image is developed from the previous model so that it can take any input image as a query (no need to be present in the dataset). The features from the query image are extracted and compared with the ones in the pickle file. The top four results are displayed on a separate window.

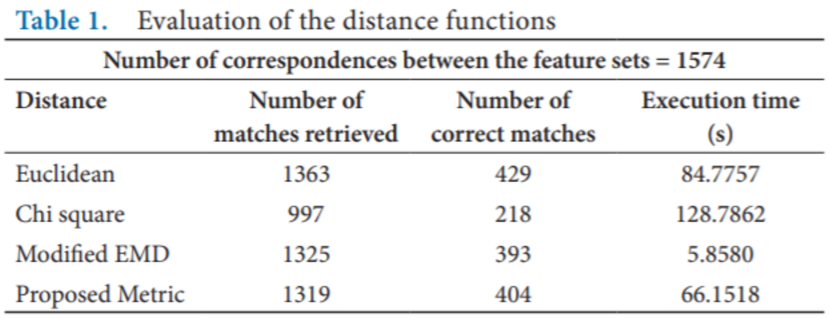


Table 1. Evaluation of the distance functions

Number of correspondences between the feature sets = 1574			
Distance	Number of matches retrieved	Number of correct matches	Execution time (s)
Euclidean	1363	429	84.7757
Chi square	997	218	128.7862
Modified EMD	1325	393	5.8580
Proposed Metric	1319	404	66.1518

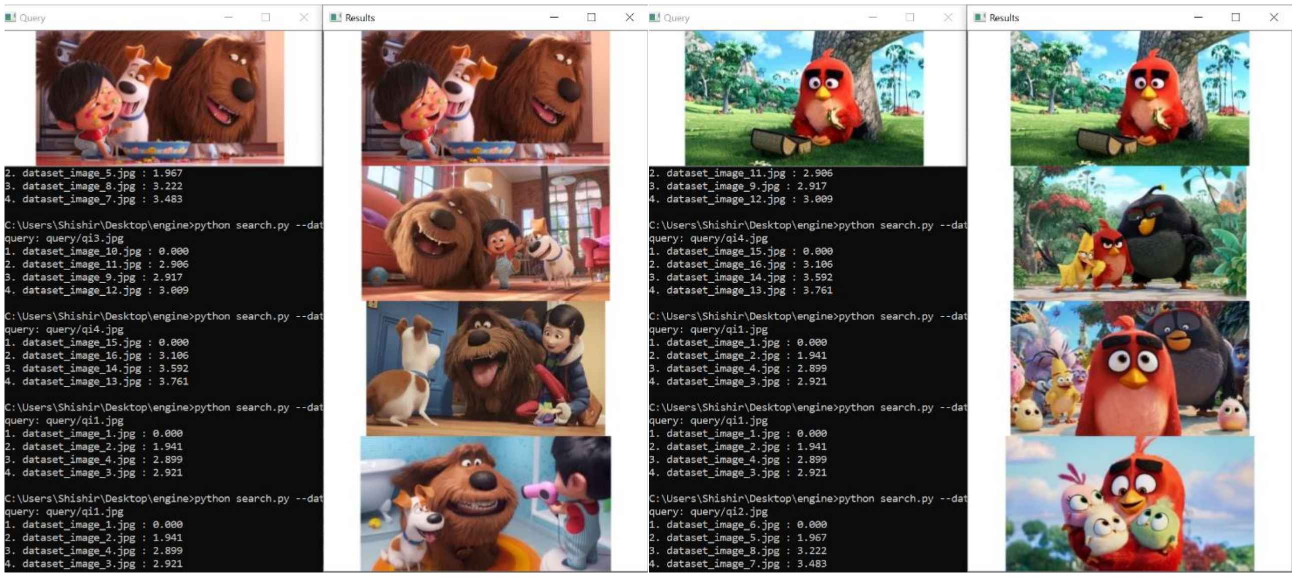
Conclusion:

It can be expected that the suggested approach of solely using the color histogram for the purpose of image representation is not enough to get a functional solid algorithm. Two dissimilar images could have similar representation if they happen to have a similar color scheme. Numerous researches in this filed is in progress to determine algorithms which will take minimum execution time and also improve the efficiency and accuracy of these systems to find the results.

Verification and testing:

Some of the screenshots while testing the images - displaying the query image and the retrieved resulting images. Both Euclidean and Chi-squared distances produced the same result but with different sorting.

Using Euclidean distance metric:



```
2. dataset_image_5.jpg : 1.967
3. dataset_image_8.jpg : 3.222
4. dataset_image_7.jpg : 3.483

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi3.jpg
1. dataset_image_10.jpg : 0.000
2. dataset_image_11.jpg : 2.906
3. dataset_image_9.jpg : 2.917
4. dataset_image_12.jpg : 3.009

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi4.jpg
1. dataset_image_15.jpg : 0.000
2. dataset_image_16.jpg : 3.106
3. dataset_image_14.jpg : 3.592
4. dataset_image_13.jpg : 3.761

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi1.jpg
1. dataset_image_1.jpg : 0.000
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi1.jpg
1. dataset_image_1.jpg : 0.000
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921
```

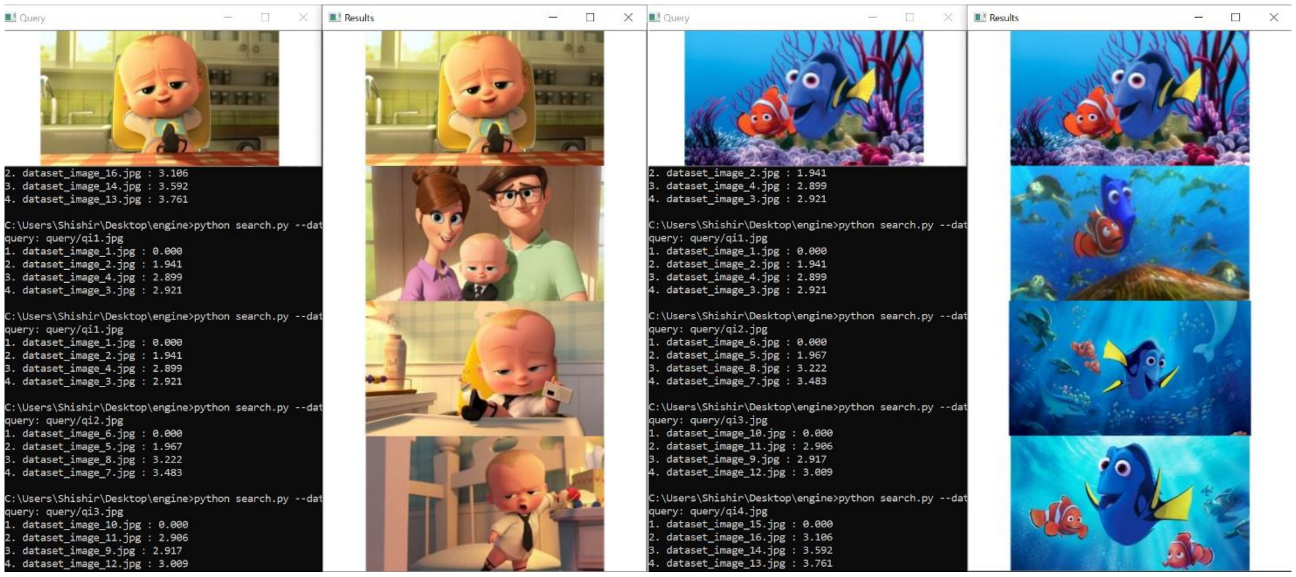
```
2. dataset_image_11.jpg : 2.906
3. dataset_image_9.jpg : 2.917
4. dataset_image_12.jpg : 3.009

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi4.jpg
1. dataset_image_15.jpg : 0.000
2. dataset_image_16.jpg : 3.106
3. dataset_image_14.jpg : 3.592
4. dataset_image_13.jpg : 3.761

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi1.jpg
1. dataset_image_1.jpg : 0.000
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi1.jpg
1. dataset_image_1.jpg : 0.000
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi2.jpg
1. dataset_image_6.jpg : 0.000
2. dataset_image_5.jpg : 1.967
3. dataset_image_8.jpg : 3.222
4. dataset_image_7.jpg : 3.483
```



```
2. dataset_image_16.jpg : 3.106
3. dataset_image_14.jpg : 3.592
4. dataset_image_13.jpg : 3.761

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi1.jpg
1. dataset_image_1.jpg : 0.000
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi1.jpg
1. dataset_image_1.jpg : 0.000
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi2.jpg
1. dataset_image_6.jpg : 0.000
2. dataset_image_5.jpg : 1.967
3. dataset_image_8.jpg : 3.222
4. dataset_image_7.jpg : 3.483

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi3.jpg
1. dataset_image_10.jpg : 0.000
2. dataset_image_11.jpg : 2.906
3. dataset_image_9.jpg : 2.917
4. dataset_image_12.jpg : 3.009
```

```
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi1.jpg
1. dataset_image_1.jpg : 0.000
2. dataset_image_2.jpg : 1.941
3. dataset_image_4.jpg : 2.899
4. dataset_image_3.jpg : 2.921

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi2.jpg
1. dataset_image_6.jpg : 0.000
2. dataset_image_5.jpg : 1.967
3. dataset_image_8.jpg : 3.222
4. dataset_image_7.jpg : 3.483

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi3.jpg
1. dataset_image_10.jpg : 0.000
2. dataset_image_11.jpg : 2.906
3. dataset_image_9.jpg : 2.917
4. dataset_image_12.jpg : 3.009

C:\Users\Shishir\Desktop\engine>python search.py --dat
query: query/qi4.jpg
1. dataset_image_15.jpg : 0.000
2. dataset_image_16.jpg : 3.106
3. dataset_image_14.jpg : 3.592
4. dataset_image_13.jpg : 3.761
```

References:

[1] A. Rosebrock, “The complete guide to building an image search engine with Python and OpenCV” [Online]. Available: https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/

[2] W. Zhou, H. Li and Q. Tian,” Recent Advances in Content-based Image Retrieval: A literature Survey”

[3] CBIR, Content-based Image Retrieval, J. M., Department of Computer Science, Aristotle University

[4] I. Zaitsev, “The Best Format to Save Pandas Data” [Online]. Available: https://towardsdatascience.com/the-best-format-to-save-pandas-data-414dca023e0d

[5] V. Vaithiyanathan et al. “Evaluation of Distance Functions for the Comparison of Gradient Orientation Histograms”, School of Computing, SASTRA University