CBIR PROJECT REPORT

Image search engine based on Edge Histogram Distance

Realized by:

BAKLOUTI EYA

BELLAKHAL Mohamed

Project supervisor:

Mr. Tebourbi Riadh

SUP'COM

Table of Contents:

1. Introduction to image search engines:

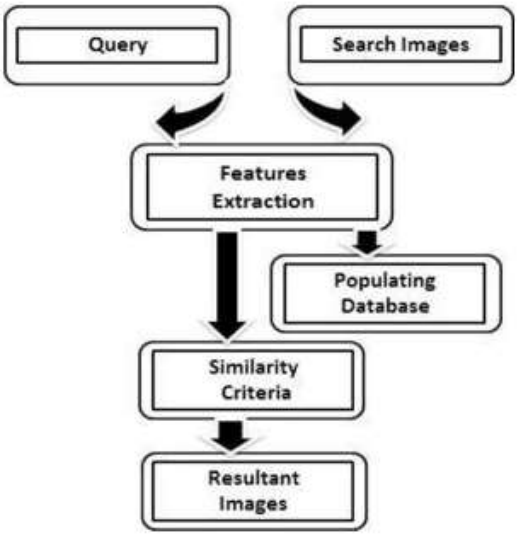
One of the most difficult tasks in the data era is to retrieve the relevant information in response to a query. To help a user in this context various search system/engine are there in market with different features.

Previously, in web search era 1.0 the main focus was on text retrieval using link analysis. It was totally a read only operation. There was no interaction in between the user and the search engine meaning that after obtaining search results the user have no option but to provide feedback regarding whether the result is relevant or not. In web search era 2.0 the focus was on retrieval of data based on relevance ranking as well as on social networking to read, write, edit and publish the result.

Content-based image retrieval (CBIR), also known as query by image content (QBIC) and content-based visual information retrieval (CBVIR) is the application of computer vision to the image retrieval problem, that is, the problem of searching for digital image in large databases. "Content-based" means that the search will analyze the actual contents of the image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. Without the ability to examine image content, searches must rely on metadata such as captions or keywords, which may be laborious or expensive to produce.

2. The Project

Our CBIR project consists of indexing a database of images retrieved from the Flickr data set and then developing a solution that finds similar images to an image uploaded through a web interface and launching a query based on the content of the images.



3. Image descriptors:

An image descriptor defines the algorithm that we are utilizing to describe our image. For example:

- The mean and standard deviation of each Red, Green, and Blue channel, respectively,
- The statistical moments of the image to characterize shape.
- The gradient magnitude and orientation to describe both shape and texture.

An image descriptor therefore governs in which way the image is quantified. Features, on the other hand, are the output of an image descriptor. When you put an image into an image descriptor, you will get a feature as an output at the other end. In the most basic terms, features (or feature vectors) are just a list of numbers used to abstractly represent and quantify images.

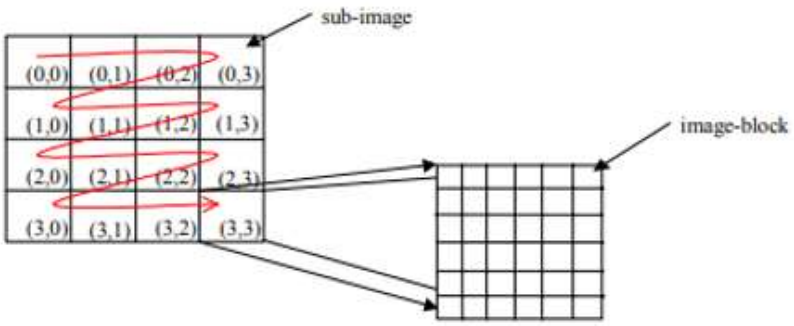
In our project we are going to use edge **histogram image descriptor**

4. Edge histogram image descriptor:

The edge histogram descriptor (EHD) is one of the widely used methods for shape detection. It basically represents the relative frequency of occurrence of 5 types of edges in each local area called a sub-image or image block. The subimage is defined by partitioning the image space into 4x4 non-overlapping blocks as shown in the figure. So, the partition of image definitely creates 16 equal-sized blocks regardless of the size of the original image. To define the characteristics of the image block, we then generate a histogram of edge distribution for each image block. The Edges of the image block are categorized into 5 types: vertical, horizontal, 45-degree diagonal, 135-degree diagonal and non-directional edges, as shown in Figure 2. Thus, the histogram for each image block represents the relative distribution of the 5 types of edges in the corresponding sub-image.

4.1. Partition of Image Space for Edge Identification and Localization:

To localize edge distribution to a certain area of the image, we divide the image space into 4x4 sub-images as shown in Figure. Then, for each sub-image, we generate an edge histogram to represent edge distribution in the sub-image. To define different edge types, the sub-image is further divided into small square blocks called image-blocks. The size and the number of image-blocks in each sub-image will be described in section 3.



4.2. Edge Types:

As shown in the Figure, five edge types are defined in the edge histogram descriptor. They are four directional edges and a non-directional edge. Four directional edges include vertical, horizontal, 45 degree, and 135 degree diagonal edges. These directional edges are extracted from the image-blocks. If the image-block contains an arbitrary edge without any directionality, then it is classified as a non-directional edge. Extraction of edge information from the image block will be described in section 3.

a) vertical edge | b) horizontal edge | c) 45 degree edge | d) 135 degree edge | e)non-directional edge

4.3. Semantics of Local Edge Histogram:

After the edge extraction from image-blocks, we count the total number of edges for each edge type in each sub-image. Since there are five different edges, we can define five histogram bins for each sub-image. Then, since there are 4x4=16 sub-images, we have total 16x5=80 bins for the edge histogram. By scanning sub-images according to the order shown in Figure 1, the semantics of the bins are defined as in Table 1.

Table 1. Semantics of local edge bins

Histogram bins	Semantics
Local_Edge [0]	Vertical edge of sub-image at (0,0)
Local_Edge [1]	Horizontal edge of sub-image at (0,0)
Local_Edge [2]	45degree edge of sub-image at (0,0)
Local_Edge [3]	135 degree edge of sub-image at (0,0)
Local_Edge [4]	Non-directional edge of sub-image at (0,0)
Local_Edge [5]	Vertical edge of sub-image at (0,1)
⋮	⋮
Local_Edge [74]	Non-directional edge of sub-image at (3,2)
Local_Edge [75]	Vertical edge of sub-image at (3,3)
Local_Edge [76]	Horizontal edge of sub-image at (3,3)
Local_Edge [77]	45degree edge of sub-image at (3,3)
Local_Edge [78]	135 degree edge of sub-image at (3,3)
Local_Edge [79]	Non-directional edge of sub-image at (3,3)

4.4. Normalization and Quantization of the Bins:

After generating local edge histograms for all 16 sub-images, we need to normalize each bin in the histogram by dividing it with the total number of image-blocks with an edge in the corresponding sub-image. Then, each histogram bin has a value ranging from 0 to 1. To represent the normalized bin values in binary form, we need to quantize them. Since the normalized bin values are normally distributed in a small range (say, from 0 to 0.3), bin values are non-linearly quantized. The quantization tables are obtained by adopting the Lloyd-Max algorithm. Then, assigning 3 bits per bin we have total 3x80=240 bits to represent the local histogram

5. KMeans:

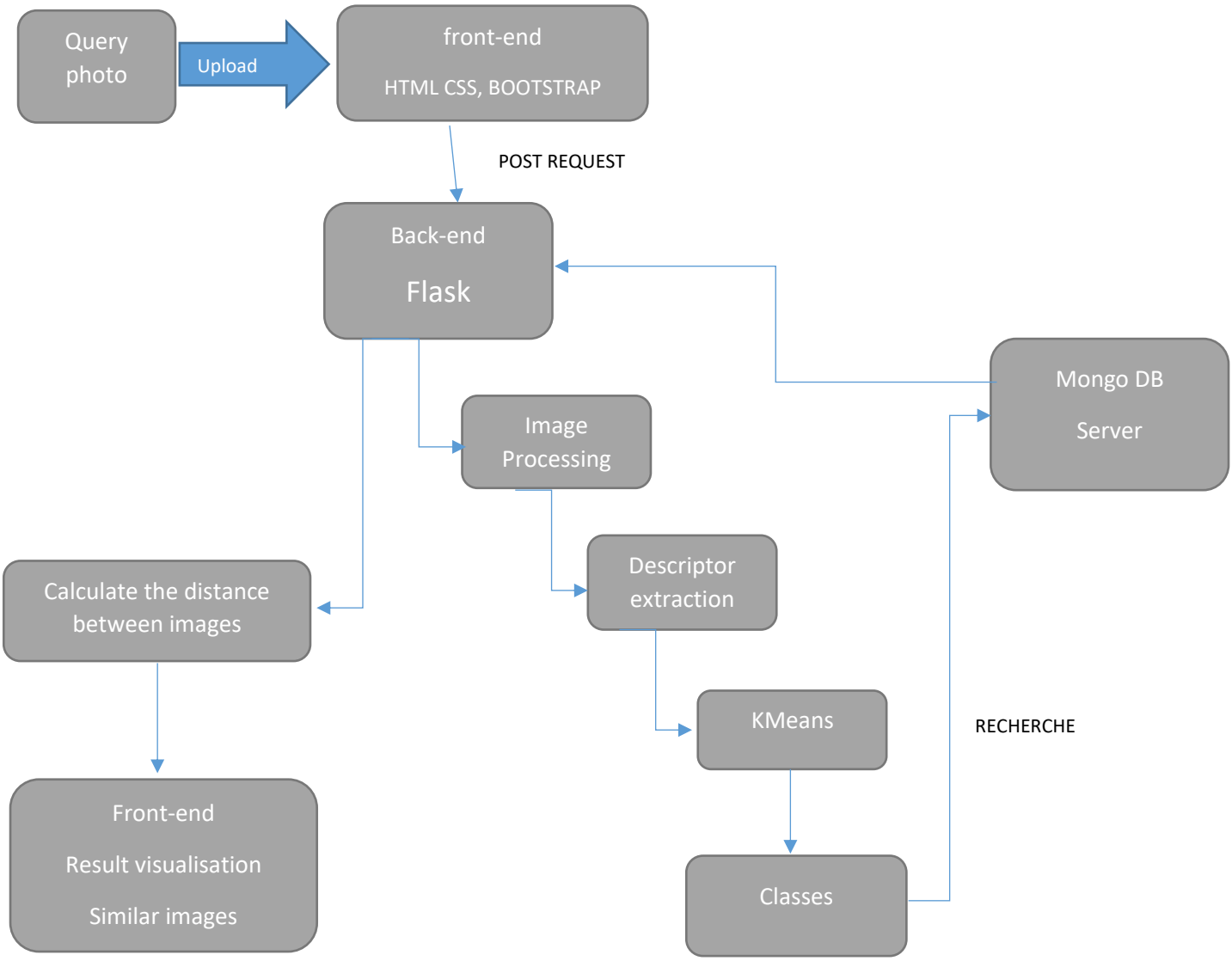
KMeans algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping sub-groups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way KMeans algorithm works is as follows:

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

6. The solution Architecture:



7. The project main steps:

1. ***Installing the image dataset and Define your image descriptor :*** Before we even consider building an image search engine, we need to consider how we are going to represent and quantify our images using only a list of numbers .We explored three aspects of an image that can easily be described: color, texture, and shape. In our project we opted to work with the MIRFLICKR 1 million images data set and its Edge histogram descriptors which are already calculated.

   http://press.liacs.nl/mirflickr/mirdownload.html?fbclid=IwAR127OzNAi0nLgABNaoI0J76LtWuoivmCtPfB4knrNenao5AKjxR8pFpprQ

   2. Thumbnail images

   3 zip files containing 100 directories with 10000 thumbnail images each

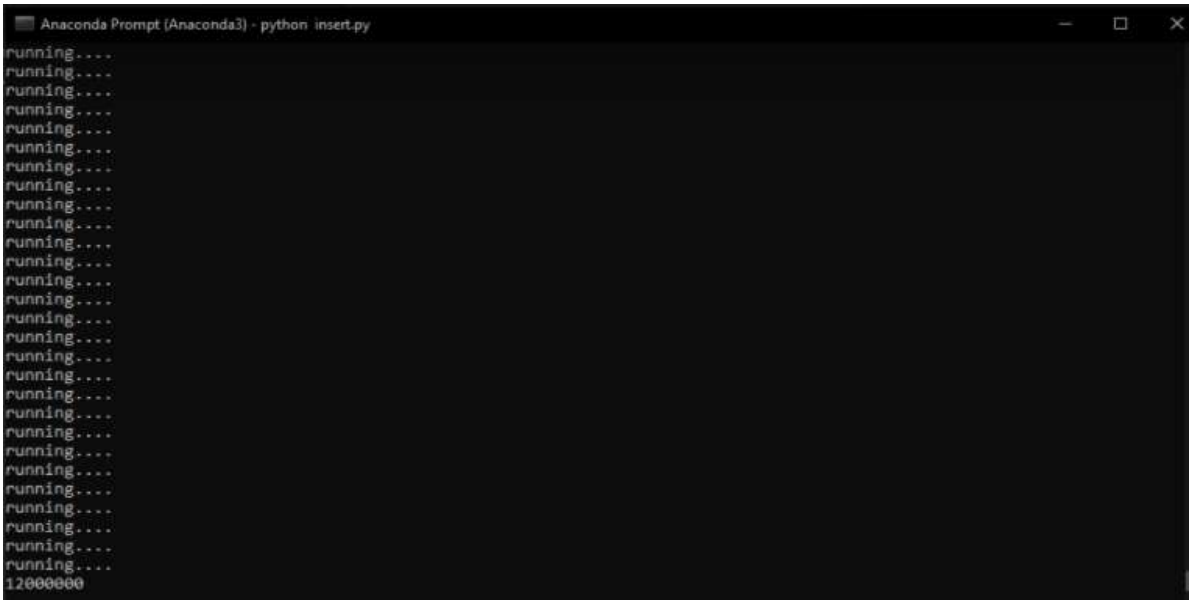
   Directories 1 - 33: thumbs1-33.zip
   Directories 34 - 66: thumbs34-66.zip
   Directories 67 - 100: thumbs67-100.zip

   4. Metadata - Visual Descriptors

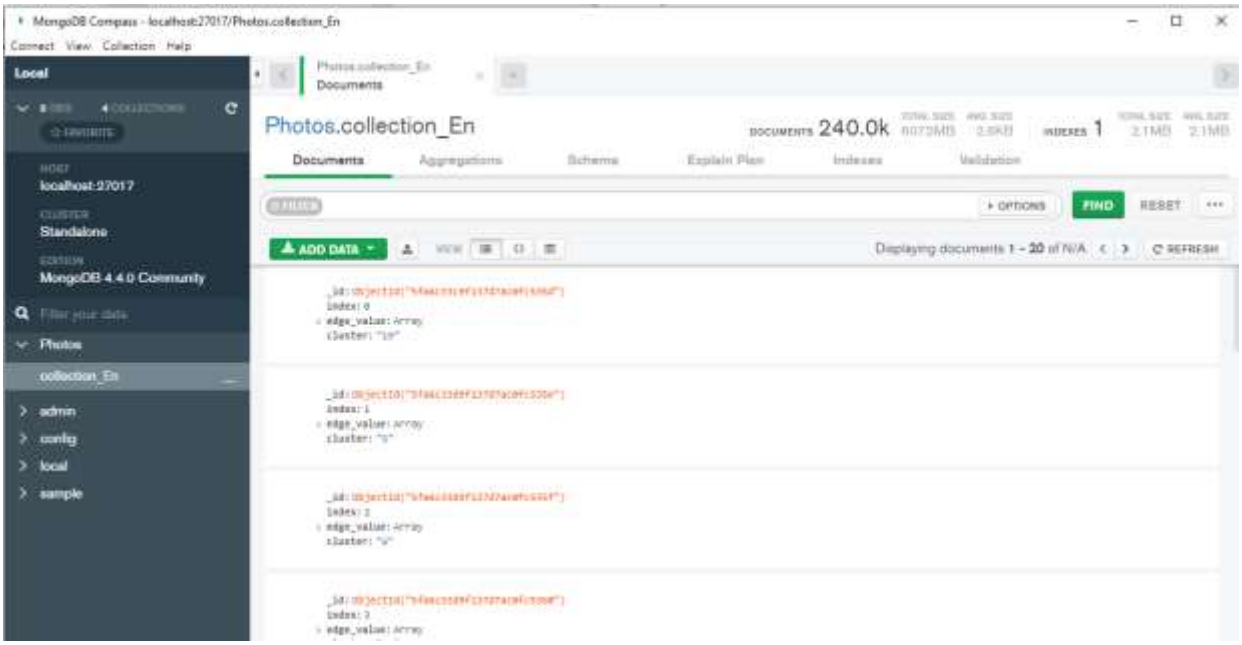
   100 files with descriptors for 1000 images each. More detailed documentation will be added shortly.

   MPEG-7 Edge Histogram descriptors: Edge Histogram Descriptors
   MPEG-7 Homogeneous Texture descriptors: Homogeneous Texture Descriptors.zip

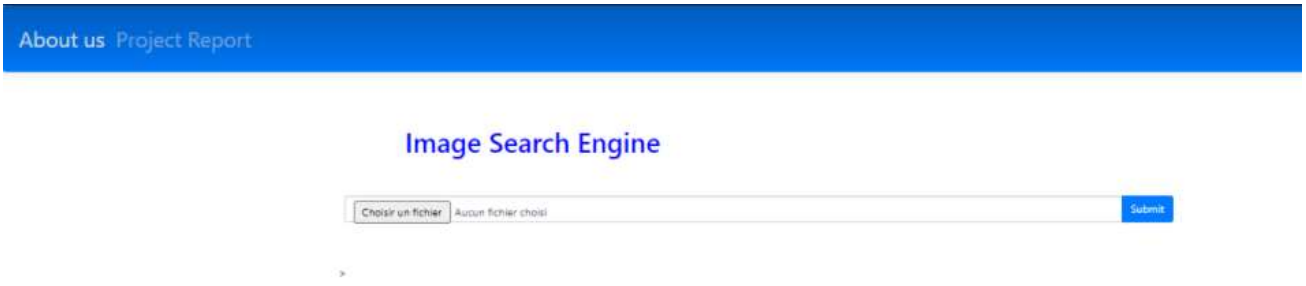
2. ***Insert the data in the MongoDB data base:*** we created a python script to insert the data that we have downloaded to the MongoDB database using the Pymongo library and cluster them using KMeans clustering algorithm which we applied on the descriptor of each photo of our data set.



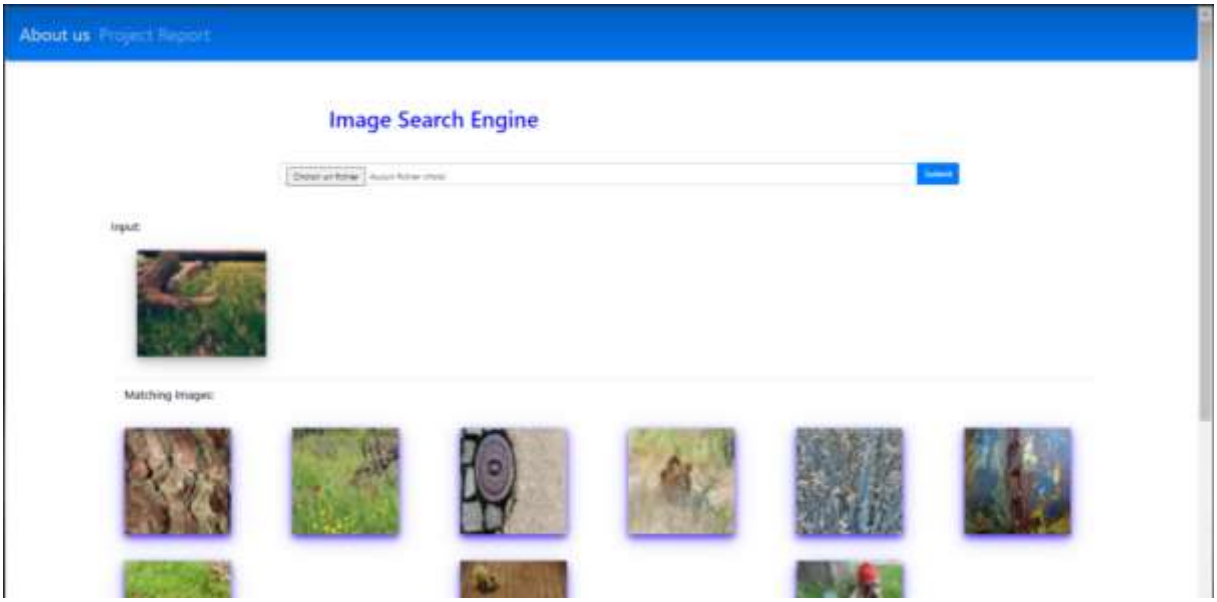
Here on the mongo dB compass we can see that our indexing of the images was successful and that the data is successfully pushed to the mongo dB server.



3. ***Create the web application*** **:** create a graphical front-end interface using HTML CSS and Bootstrap in which the user can upload the photo that he want to find the similar ones to it and develop a backend application app.py using Flask .



4. ***See the result*** : run app.py and find in the local host the result and try the search engine



8. Conclusion

- This project covered several different topics such as computer vision, artificial intelligence, database management, Rest APIs and web development as it was a great opportunity to apply the image processing skills that we learned through this entire course as well as getting more insight on the other technologies mentioned above.

- In this project we weren't able to work on the 1 Million photos offered by Flickr due to hardware limitations therefore we opted instead to work on 100 000 photos and their respective descriptors