Content Based Image Retrieval: Importance and Applications

İdil Sülo
Department of Computer Engineering
Middle East Technical University
Ankara, TURKEY 06800
Email: idil.sulo@ceng.metu.edu.tr

***Abstract*—This paper presents a brief analysis of the methods used in CBIR - Content Based Image Retrieval while underlying the importance of this technique due to the growth of the Internet which causes the accumulation of large collections of data. For the experiment, we developed several applications of this technique over an image database using two well-known algorithms to describe local features in images named SIFT - Scale Invariant Feature Transform, and Dense-SIFT - Dense Scale Invariant Transform. Also, we have experimented these implementations using different number of clusters.**

***Index Terms*—Content Based Image Retrieval, Interest Points, Local Feature Descriptors, SIFT, Dense-SIFT, Clustering, Bag of Feature Representation**

I. Introduction

In the age of big data, data sets grow rapidly due to the increase of data gathered by numerous information-sensing internet of things devices. Considering this scenario, it is rather important to efficiently manage these data collections by using convenient information systems. The most common approach to manage these collections is called CBIR- Content Based Image Retrieval systems.

The goal of CBIR system is to search for images that are semantically highly similar to a given query image, within a large database. The search is performed by comparing the representation of the query image to the images that reside in the database. The representation of each image is based on the content properties such as the shape, color or the texture of the image that are encoded into feature vectors. The potential uses of CBIR systems include crime prevention, face finding and medical diagnosis among many other areas.

The aim of this paper is to point out the importance and describe the applications of Content Based Image Retrieval system. We focus on the implementation of CBIR to analyze the results based on the use of different approaches to extract local descriptors from images, and the use of different number of clusters.

The article presents the importance of Content Based Image Retrieval and the motivation behind this system in Section II, whereas in Section III, it describes the CBIR system in detail, and explains our implementation. Section IV presents the two different algorithms for feature extraction to obtain image representation, whereas Section V discusses the effect of assigning different number of clusters. In Section VI, we explain the ranking of the images with respect to a similarity test. Section VII discusses our experiments and evoluation of a few query results in the CBIR system. Section VIII states our conclusions.

II. Motivation

The amount of digital image collections have grown enormously with the rapid increase of the online users on the Internet and the web applications that allows the users to add images and digital albums. In addition to that, images are now globally used to gather information in several areas such as weather prediction and advertising which also contributes to this growth.

Complexity of image data and its interpretation is also another problem that resides with the previous one. Therefore, to manage these large and complex collections of digital images effectively, it is important to develop appropriate systems in the image retrieval area. [1]

III. Content Based Image Retrieval

In this section, we briefly describe the Content Based Image Retrieval method, and present its implementation in the Python programming language.

A. Description

The goal of Content Based Image Retrieval is to find similar images to a given query image by comparing them with respect to their color, texture or shape and measuring their similarity.

At first, CBIR pre-extracts the features of each image in its database. Then, when a new input query image is provided, it also extracts features for this query image. Next, the similarity between the representations of the query image and the images in the database is measured. Lastly, the CBIR system outputs the most similar images to the given query image from its database.

The pipeline of the CBIR system is shown in Fig.1.

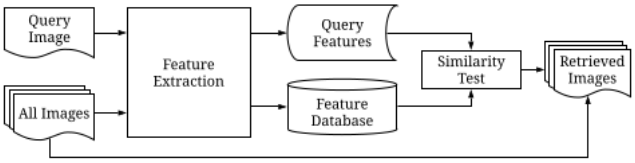

Fig. 1. CBIR system pipeline

B. Implementation

For the development of Content Based Image Retrieval system, Python programming language is chosen. We have used Python3 with Anaconda distribution.
The procedure that is used for the implementation of this system can be explained as follows:

1) Convert all images in the dataset to grey scale.
2) For each image in the dataset, apply SIFT or Dense-SIFT algorithm to extract local descriptors to be stored as numpy arrays in the image object with shape (N,128) where N is the number of keypoints in each image. Next, concatenate all these local descriptors with a descriptors vector to store each of them together.
3) To construct a visual dictionary, apply k-means algorithm over the descriptors vector mentioned above, which stores the information of all local descriptors in each image. Then, store the clusters returned from the k-means algorithm into a pickle file named *clusters.pkl*. Note that K value for the k-means algorithm is given to the script file while execution, where K represents the number of cluster centers and takes one of the values from the following set: {32, 64, 128, 256}.
4) To extract the Bag of Features representation of each image, calculate the histogram of its visual words where each bin of the histogram corresponds to a cluster center that can also be noted as visual word.
5) Normalize each histogram by dividing with the number of visual words in that image, then store each of those vectors into the corresponding image object.
6) Store the image list consisting of all image objects into a pickle file named *imagelist.pkl*.
7) Convert each query image into grey scale.
8) Apply above steps for each query image without storing each of their local descriptors, to a descriptors vector.
9) Use Euclidian distance to output the distance between the BoF feature vectors of each query image and the images in the dataset.

Since the dataset that we work with includes 1491 images, it is costly to construct feature vectors for each time a query image is requested. Therefore, we have decided to extract feature vectors for each image in the dataset only once, and store these values into a file. For this purpose, we have used *pickle* module of Python which implements binary protocols for serializing and de-serializing a Python object structure. *Pickling* is the process whereby a Python object hierarchy is converted into a byte stream, and *unpickling* is the inverse operation, whereby a byte stream. [2]

IV. Local Descriptor Extraction

A. SIFT

SIFT is an algorithm proposed by David Lowe which solves the viewpoint change, intensity, affine transformations and image rotation problems in image feauture extraction. The SIFT algorithm applies four steps which can be explained as following:

1) It estimates a scale space extrema using the Difference of Gaussian (DoG).
2) It localizes key points where the key point candidates are refined by eliminating the low contrast points in the image.
3) It assigns orientation to each keypoint based on the local image gradient.
4) It computes local image descriptor for each keypoint based on the image gradient orientation and magnitude.

B. Dense-SIFT

Dense-SIFT algorithm defines patches at regular intervals using a spatial grid rather than using Difference of Gaussians (DoG) to extract SIFT descriptors for each local patch.

V. Bag of Features Representation

This representation is generated basically by constructing a visual dictionary. After running k-means algorithm for the local descriptors of all images each cluster center is considered as a visual word. The collection of these visual words are then considered as a visual dictionary. After this step, for each step we have computed the histogram of its visual words where each visual word is represented by a bin of the histogram. It is important to note that the number of visual words may vary for each image since the number of key points is not the same in each of them. Therefore, each histogram is normalized by l_1 normalization scheme which is simply dividing each histogram by the total number of visual words in that image.

VI. Ranking Images by Similarity

To compare the BoF feature vector of the query image to the BoF feature vector of each image in the dataset, Euclidean distance is used. Euclidean distance between to vectors x and y can simply be denoted as follows:

$$d_{euclidean}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i^2 - y_i^2)}$$

Then, the images are sorted by their similarity to the query image where the image which is ranked the first for the query image has the least Euclidean distance value. All distance values between the query images except the distance value of the query image to itself (which has the value of zero) is

recorded to an output file named *result.out*.
This result is then evaluated by a Python script named convert_for_eval.py to give a rank to each distance metric. Then, a ground truth file and the output from this Python script is given to another script file, namely, compute_map.py to obtain the Mean Average Precision (MAP) score of the output.
Note that, MAP metric is obtained by the mean of the average precision (AP) values, where AP is the ratio given by the following:

$$\frac{\text{number of true positives}}{\text{number of false positives} + \text{number of false negatives}}$$

VII. Experiments and Results

In our Content Based Image Retrieval system experiments, we used a dataset of 1491 color images. All the images in the dataset have dimensions 640x480 or 480x640.

We experimented with a different range of functions. For the SIFT experiments, *SIFT_create* from *cv2* library is used, and for the Dense-SIFT experiments, *dsift* from the *cyvlfeat.sift* library is used. To apply k-means algorithm, we have used *kmeans* and *MiniBatchKMeans* from the *sklearn.cluster* library.

First of all, we started our experiment using *SIFT_create* and *kmeans* function. We did not give any parameters to the *SIFT_create*, therefore we observed a huge performance gap. Then, we decided to set the *nfeatures* parameter to 100, to observe the difference. Below, are the k values and the MAP scores that we get from this experiment:

TABLE I
Table of K-means values vs. MAP scores

K Value	MAP score
128	0.20784
256	0.21350

Since the MAP scores are not quite good, we decided to experiment by setting *nfeatures* parameter to 1000. The results we obtained are as the following:

TABLE II
Table of K-means values vs. MAP scores

K Value	MAP score
32	0.31642
64	0.36578
128	0.39324

Unfortunately, we were not able to experiment with the k value for 256 since the experiment took more than 5 hours. Therefore, we decided to change our implementation by using *MiniBatchKMeans* for the k-means algorithm. The results from this experiment were not as accurate as we did before, but it reduced the performance gap more than it is expected. The k-means values and their corresponding MAP scores for the experiment with *MiniBatchKMeans*, and *SIFT_create* where the *nfeatures* parameters in this function equals 1000 is given as the following:

TABLE III
Table of K-means values vs. MAP scores

K Value	MAP score
32	0.30833
64	0.36321
128	0.37420
256	0.41281

The results that are presented above are plotted using *matplotlib* library of Python as following:

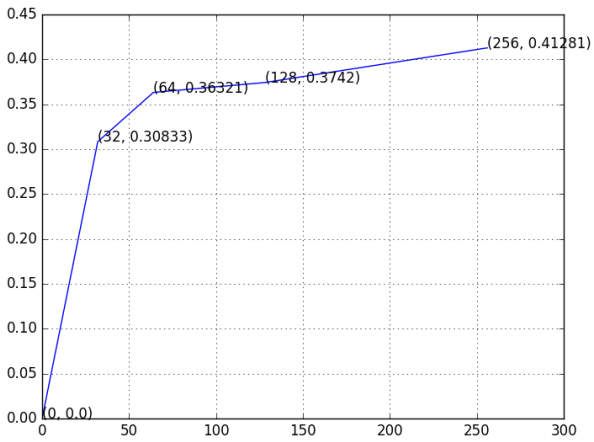

Fig. 2. K values of k-means algorithm

An example query for this experiment can be given as:

Fig. 3. Query image

The most similar images found by the CBIR system corresponding to above query when it runs with k value equals 256 can be shown as:

Fig. 4. Result image

Fig. 5. Result image

As the last experiment we run the CBIR system using *dsift* to apply Dense-SIFT algorithm. Since there is a huge performance gap compared to SIFT algorithm we were only able to run the CBIR system with the k value 32 for k-means algorithm. The results that we have obtained are as follows:

TABLE IV
TABLE OF STEP VALUES VS. MAP SCORES

K Value	MAP score
100	0.02502
10	0.37279

For the above table, *step* is an attribute that is given to the *dsift* function. The default value of it is 3.

To sum up, the CBIR system get the performance for the k value 256 for the k-means algortihm, and SIFT algorithm for feature extraction. It is more accurate when it uses Dense-SIFT algorithm but the the performance drops for that algorithm.

VIII. Conclusion

By the growth of the Internet and the web applications the amount of digital images have grown continuously which caused the development of effective systems in the image retrieval area to efficiently manage these data collections. Systems that corresponds to managing the large collections of visual data is known as Content Based Image Retrieval systems.

This paper presented an overview on Content Based Image Retrieval. In particular, it uses two different algorithms for extracting feature vectors which are SIFT and Dense-SIFT. Then, we used k-means algorithm to construct a visual dictionary, and compared the query images with each image in the datatbase using Euclidean distance.

As mentioned in this paper, it is important to use the appropriate combination of the algoritms and parameters in order to optimize the performance of the CBIR system. As future work, the system can be implemented on a machine with higher performance to get more accurate results in an optimized response time.

References

[1] J. Junior and R. W. Marcal and M. Batista, *Image Retrieval: Importance and Applications*, 2014.

[2] 12.1. pickle - Python object serialization. (n.d.). Retrieved April 12, 2018, from https://docs.python.org/3/library/pickle.html

[3] Ebrahim Karami, Siva Prasad, and Mohamed Shehata *Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images*, Memorial University, Canada