GIF Image Retrieval in Cloud Computing Environment

Evelyn Paiz-Reyes(✉), Nadile Nunes-de-Lima(✉), and Sule Yildirim-Yayilgan(✉)

Norwegian University of Science and Technology, Gjøvik, Norway
{elreyes,nadilend}@stud.ntnu.no,
sule.yildirim@ntnu.no

Abstract. GIF images have been used in the last years, especially on social media. Here it is explored a content-based image retrieval system to work specifically with GIF file format. Its implementation is extended to a cloud computing environment. Given the Tumblr GIF dataset, it is created a "search by example" image retrieval system. To describe the images, low-level features are used: (1) color, (2) texture and (3) shape. The system performs the search using just GIF images as query images. To obtain faster results on the retrieval process, a hashing indexing approach is used. The system showed a complexity of $O(n^2)$ for indexing and $O(log(n))$ for retrieval. Additionally, better results were obtained (in relation to precision and recall) for simple images, instead of images with a lot of movements.

Keywords: GIF image format · Content-based image retrieval
Hashing · Cloud computing

1 Introduction

The importance of images in people's daily life has increased along with the advances in digital technology. The emergence of imaging devices, such as smartphones, cameras, and computers, has led the world to witness the growth in quantity, availability, and value of images [1]. The need to access this type of data anytime and anywhere has increased its importance.

Content-based image retrieval (CBIR) has been widely studied [2–4]. A CBIR system retrieves the relevant images from a database to a search query based on visual characteristics, where each image in the database is mapped to a feature vector including visual, structural and conceptual characteristics [5]. On the other hand, a CBIR maintenance is considered to be a typical example of cloud computing. It offers a great opportunity for on-demand access to a wide computation and storage resources [1].

Among the many image formats that can be applied on a CBIR system, one that is interesting to mention is the GIF (Graphics Interchange Format) file format. Because of its high compression ratio and less disk space needs, this image format has been spread quickly and supported by a wide variety of applications [6]. It was designed "to allow the easy interchange and viewing of image data stored on local or remote computer systems" [7].

GIF has become well known and it is the second most common image format used on the World Wide Web after JPEG. Up until now, there have been few attempts to explore the design and development of a CBIR system for this file format. The objective of the present work is first to design and develop a CBIR system for indexing and retrieval of GIF file formats using the Tumblr GIF dataset [8]. Secondly to extend the CBIR to a cloud computing environment.

2 Related Works

2.1 Content-Based Image Retrieval System

No matter what type of CBIR system is intended to be built, the following steps are the ones needed to complete it [1,9,10]:

Image Descriptors. An image feature can be described as anything that is localized, meaningful and detectable from an image [4]. In practice, the use of multiple features is needed to be able to achieve a good description of the image. The features/signatures intended for this work are color [4], texture [4,10] and shape [4].

Dataset Indexing. Indexing can be described as the process of quantifying the dataset by utilizing an image descriptor to extract features from each image.

Definition of Similarity Metric. Images can be compared using a distance metric or similarity function. This function takes the feature vectors as inputs and its output is a number representing how "similar" the feature vectors are. The choice is highly dependent on (1) the dataset and (2) the types of features that were extracted.

Search. The search process is the following: (1) A user submits a query image to the system, (2) its features are extracted and (3) the similarity function is applied to compare the query features with the features already indexed. From there, (4) the results are sorted by relevancy and then presented to the user.

2.2 CBIR and Graphics Interchange Format (GIF)

GIF is a standard image representation and has the possibility to create animations, which made it popular on social media. There is no work related to a CBIR system specific for indexing and retrieving GIF files. The GIF format was developed in 1987 by Steve Wilhite at CompuServe Network [6]. Its purpose is to store multiple bitmap images in a single file for exchange between platforms and images. It is stream based and is made up of a series of data packets called blocks and protocol information. In GIF, lossless file compression method is applied [7].

GIF file format comprises a number of frames that are displayed in succession. Each frame is displayed with a time delay to wait until the frame is drawn [6]. Then, *why not call a GIF a video?* Because a video is a collection of both inter and intra coded frames. In a video, the inter-coded frames take advantages of the other frames, so compression is far more efficient. On the other hand, a GIF is just a collection of intra frames and each frame is coded on its own [11].

3 Methodology

The goal is a GIF-based CBIR system. A GIF is an image file and not a video file [11], but it has the characteristic of frames as video files do. Therefore, this is considered in the design of the system (Fig. 1).

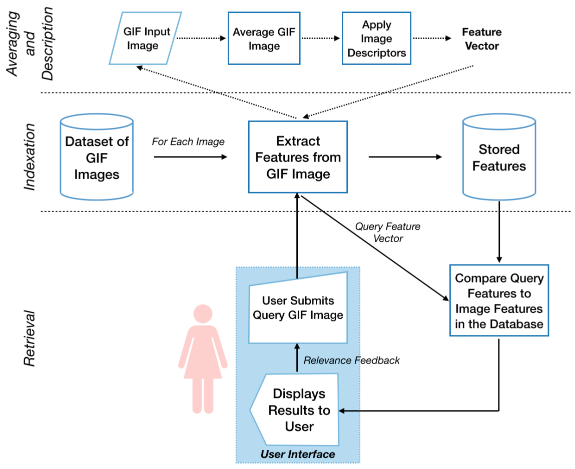


Fig. 1. Flowchart representing the architecture of the designed system.

3.1 Definition of Dataset

A collection of 2k randomly selected GIF images from the Tumblr GIF dataset (containing 100k animated GIFs in total) is used. The dataset consists of various GIFs images from Tumblr taken from random posts between May and June of 2015 [8].

Because the dataset was not previously divided into classes, the collection was manually classified into four categories where the images belonging to each category were considered similar (subjectively by the experimenter). It is important to mention that it was not possible to categorize all images. The dataset was composed of categorized plus uncategorized images. The final classes were the following:

1. Grayscale with couples.
2. Grayscale focused on faces and/or mainly just one person throughout the image.
3. Sports on a field (golf, soccer, baseball, etc.)
4. Animals.

3.2 Definition of Image Descriptor

The most basic image descriptors (color, texture, and shape) are employed. For each descriptor, a specific feature vector is created and then the three merged into a single feature vector. This means that each image in the dataset is represented and quantified using only a list of floating point numbers. Figure 2 shows this process. Additionally, the ad-hoc values selected were the ones that gave better result concerning the visual perception of the image and processing time.

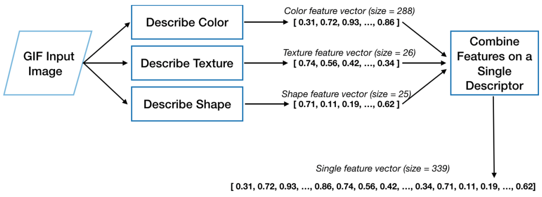


Fig. 2. Flowchart representing an example of the feature extraction.

Color-Based Descriptor. The color-based image descriptor is a 3D color histogram in the HSV color space. It is used 8 bins for the hue, 12 for the saturation, and 3 for the value, yielding a feature vector of dimension $8 \times 12 \times 3 = 288$ [10].

Texture-Based Descriptor. The method of local binary patterns (LBP) [12] is applied. First, the image is converted to grayscale and, for each pixel, a circular neighborhood of $p = 24$ points and radius $r = 8$, surrounding the center pixel is selected. An LBP value is calculated for each center pixel and stored in an output 2D array [13]. Its computed histogram is the texture-based image descriptor [12]. A total feature vector of dimension $24 + 2 = 26$ is obtained.

Shape-Based Descriptor. Zernike polynomials are orthogonal to each other and there is no redundancy of information between moments [14,15]. First, the image is converted to grayscale and segmented. Next, the Zernike moments are applied (with $r = 21$, where r is the radius of the polynomial) to characterize the shape of the object. The result is a vector with the first 25 Zernike moments from the segmented image [16].

3.3 System Architecture

The system is divided into three basic modules: (1) averaging and description of the images, (2) indexing of the data and (3) retrieval from a query. Figure 1 shows a graphical representation of the final architecture.

Averaging and Description. This module takes as an input a GIF image. A GIF image has the property of containing various images (frames) inside, and instead of working with all these data, an average image is obtained (see Algorithm 1). Next, it is computed the three feature vectors described in Sect. 3.2. The output is the combination of these vectors in a single feature vector.

Algorithm 1. Proposed averaging GIF image algorithm.

```
  Data: List of frames/images in the GIF file
1 Set sum_image to an empty image;
2 for each frame in the GIF image do
3     add sum_image with the frame image pixel by pixel;
4 end
5 average_image = (sum_image / total number of frames);
6 Return average_image;
```

Indexing. The basic process of indexing is done as in a normal CBIR (features of each image are extracted and stored), but with the inclusion of averaging. For each image of the dataset (Sect. 3.1), the average is calculated and features are extracted. The resulting feature vectors are saved in a hash table in the form of a dictionary, where information of the image (name), a feature vector and hashed value of the feature vector are stored. The method used for indexing is the locality sensitive hashing (LSH) [17], aiming to maximize the probability of a "collision" for similar items.

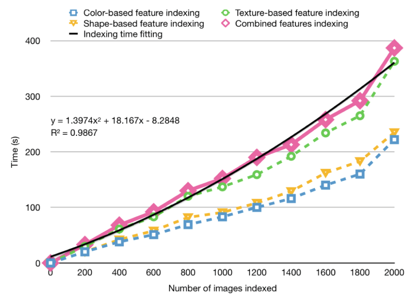
Retrieval. The module works in the following way: (1) The input is a query GIF image; (2) the image is averaged and features are extracted to a single vector; (3) The query feature vector is compared with the stored feature vectors using Euclidian distance and a set of similar results is obtained; and (4) The top 5 results are retrieved and displayed to the user.

4 Experiments and Results

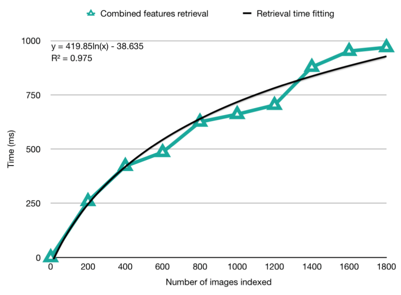
4.1 Indexing and Retrieval Time

In steps of 200 images, both indexing and retrieval time were tested (Fig. 3). The indexing time of each one of the image descriptors (individually) was also measured. A PC (Intel Core i5; 2.5 GHz) with macOS High Sierra and Python

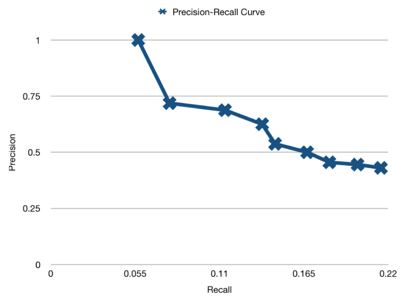
2.7 was used in the experiment. The result data (from the measured time) was fitted on a curve to obtain the complexity of both indexing and retrieval. For indexing, all image descriptors (individually and combined) showed a polynomial behavior. In retrieval, the result was a logarithmic structure instead. The final outcome of the experiment revealed a complexity of $O(n^2)$ for indexing and $O(log(n^2))$ for retrieval.



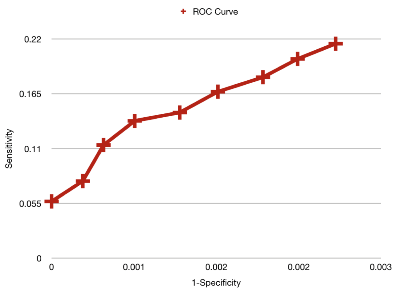
(a) Indexing time.



(b) Retrieval time.



(c) Precision and recall curve.



(d) ROC curve.

Fig. 3. Measuring curves of the system from 0 to 2000 images.

4.2 Precision and Recall of the System

On the second experiment, the precision and recall of the system were measured (Fig. 3). A total of 4 images from each category were selected randomly as query and the values of precision and recall were measured for each one of the queries (retrieving 1 to 10 images as a result). The average results were computed for each category. A final average of all the categories was the resulting precision and recall curve. Finally, a ROC curve was also computed. The same measurements of the precision-recall curve were used. Instead of estimating precision and recall, it was computed sensitivity and specificity. Figure 3 shows the resulting curves.

4.3 User Happiness

The final experiment was a user happiness evaluation. The study was performed with 6 adult volunteers from the Norwegian University of Science and Technology (NTNU). For each category (same as Sect. 3.1), a set of four images was shown. The user selected one each time and for the retrieved GIFs, the participant indicated what was considered relevant and irrelevant to him/her.

It was mentioned from their point of view that the time response for the retrieval was acceptable (83%). Also, users considered the categories 1 and 2 as similar. Finally, during the classification of what was relevant and irrelevant for the user, the semantics of the files had a strong influence. The participant tended to consider a file relevant when it contained the same actions or emotions as the query (i.e. if the query contains a person crying, the relevant results will be GIFs with people sad or crying as well).

5 Discussion and Conclusion

The complexity of the system was divided in two: indexing and retrieval. When the feature extraction is evaluated separately, it is clear that the texture description is the process that takes the majority of the time. This is because on the LBP method every single pixel in the image is computed, increasing the time needed to extract the features. Therefore, for future improvement of the system, a new method that is able to reduce the complexity of describing texture is advise.

The relevance/irrelevance of the files retrieved was judged by users and similarities between the answers were found. However, GIFs nowadays are also part of the social interaction, being used to transmit emotions and opinions towards some subject online. Therefore, only low-level features might not be enough to find the relation between the content of the file and what is the meaning the user is looking for.

The best results of the system were related to queries that did not have many motion through the images inside the file. The improvement of this condition can be a valid future work for this project. The insertion of semantics and classification would be also recommended, but first, a precise and detailed description must be done for the dataset (classifying every image). Hence, this classification would allow the use of machine learning to improve the retrieval process.

References

1. Xia, Z., Wang, X., Zhang, L., Qin, Z., Sun, X., Ren, K.: A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. IEEE Trans. Inf. Forensics Secur. **11**(11), 2594–2608 (2016)
2. Zouaki, H., Abdelkhalak, B.: Indexing and content-based image retrieval. In: 2011 International Conference on Multimedia Computing and Systems, pp. 1–5, April 2011

3. Rashno, A., Sadri, S.: Content-based image retrieval with color and texture features in neutrosophic domain. In: 2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA), pp. 50–55, April 2017
4. Kaur, M., Sohi, N.: A novel technique for content based image retrieval using color, texture and edge features. In: International Conference on Communication and Electronics Systems (ICCES), pp. 1–7, October 2016
5. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: ideas, influences, and trends of the new age. ACM Comput. Surv. **40**(2), 5:1–5:60 (2008)
6. Zhang, Y.: The studies and implementation for conversion of image file format. In: 2015 10th International Conference on Computer Science Education (ICCSE), pp. 190–193, July 2015
7. Tiwari, N., Shandilya, D.M.: Evaluation of various LSB based methods of image steganography on gif file format. Int. J. Comput. Appl. **6**(2), 1–4 (2010)
8. Li, Y., Song, Y., Cao, L., Tetreault, J., Goldberg, L., Jaimes, A., Luo, J.: TGIF: a new dataset and benchmark on animated GIF description. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016
9. Yang, Z., i. Kamata, S., Ahrary, A.: Nir: content based image retrieval on cloud computing. In: 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, vol. 3, pp. 556–559, November 2009
10. Rosebrock, A.: The Complete Guide to Building an Image Search Engine with Python and OpenCV (2014). https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/. Accessed 19 Sep 2017
11. Hu, W., Xie, N., Li, L., Zeng, X., Maybank, S.: A survey on visual content-based video indexing and retrieval. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **41**(6), 797–819 (2011)
12. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
13. Rosebrock, A.: Local Binary Patterns with Python & OpenCV (2015). https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/. Accessed 6 Nov 2017
14. Chaumette, F.: Image moments: a general and useful set of features for visual servoing. IEEE Trans. Rob. **20**(4), 713–723 (2004)
15. Kim, W.Y., Kim, Y.S.: A region-based shape descriptor using Zernike moments. Signal Process. Image Commun. **16**(1), 95–102 (2000)
16. Rosebrock, A.: Building a Pokedex in Python: Indexing our Sprites using Shape Descriptors (Step 3 of 6) (2014). https://www.pyimagesearch.com/2014/04/07/building-pokedex-python-indexing-sprites-using-shape-descriptors-step-3-6/. Accessed 6 Nov 2017
17. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 253–262. ACM (2004)