GIF Image Retrieval

Burak Enes Beygo
n19134235@cs.hacettepe.edu.tr
Hacettepe University
Ankara, Turkey

Abstract

In the last years, the popularity of GIF images have been raised. In this paper, I present an image retrieval system especially for GIF file format. This system can be used to search a GIF image inside a storage consisting a set of GIF images. The Red(R), Green(G) and Blue(B) color levels play an important part describing the features of the images in this system. Some kind of a compression algorithm is used to index the representation of images. Finally, the system returns the resemblance result after making the comparison calculations.

Keywords: image retrieval, GIF image format

1 Introduction

People use images in their digital devices excessively. Importing and exporting any data as image is so practical because an image is easily understandable for human eyes. Thus, representing data as image plays an essential role in digital world.

There are different types of image formats. Best known image format is JPEG which is an acronym for Joint Photographic Expert Groups. JPEG images can store huge numbers of colors and they have smaller file sizes compared to the other image formats, but they have disadvantages such that they do not support transparency, and some pixel information gets lost when the image is resized. On the other hand, PNG image format is used to compress image without losses, but for PNG images, file size is generally bigger than JPEG formatted images. I focused on GIF image format which is an acronym for Graphics Interchange Format. A GIF image can display 256 colors and support lossless compression. GIF images are generally used for web graphics, and it is the best format for playing animations. GIF image is comprised by a set of images.

Retrieval of images is a common topic in information retrieval area. There are different kinds of techniques for image retrieval in general, but very few for specifically GIF image format. GIF images look like a video, but video storing is completely different from the images, because videos have different components such as time and sound. In this research, I propose a method to retrieve GIF images. I used recent image retrieval researches to create this work. My novelty of work is that this research is specifically for GIF image formats. Normally, a GIF image has to be low resolution because the file size would be very high if the resolution was set as high. The main objective of this work is to accomplish how to retrieve similar GIF formatted image when one GIF image is given.

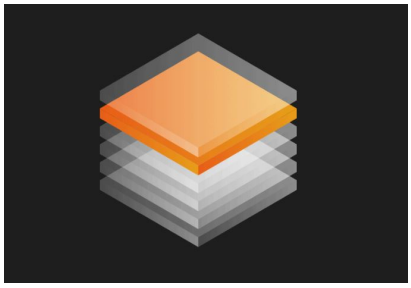

Figure 1. Images inside a GIF image (frames in some cases)

2 Related Research

Nearly all image retrieval techniques use image descriptors. An image descriptor is a feature that transformed to generate data for computer programs to understand. After that, the data in a dataset has to be indexed by extracting features from images. To compare the images, we have to make some similarity calculations. There are different types of similarity metrics are proposed in this area, but the most well-known method is the Euclidean distance. Euclidean distance is the straight line distance between two points in Euclidean space. According to my research, there is not any method or technique proposed especially for GIF formatted images. My aim is to propose a new technique consisting Euclidean distance [2]. After the calculations of similarity, we have to scan the dataset folder and make the calculations on each image. Only this way, we can find the correct result.

In some papers, people call images inside a GIF image "frames". I prefer to call them images, because they are actually basic images. There is not any paper that call GIF images videos, because videos contains other kinds of context such as time and sound. There are different kinds of retrieval techniques for videos.

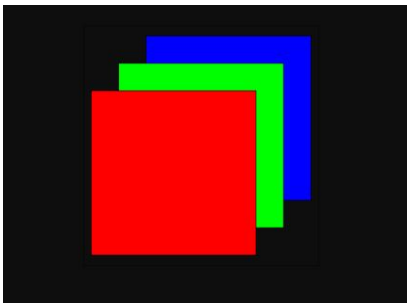

Figure 2. Color levels of an image

A recent study proposed an image retrieval system which uses neural network features as local feature descriptors [3]. They had specifically worked on landmark dataset. Their main purpose is to accomplish more accurate feature matching and geometric verification. their system successfully rejects false positives of the classified data. They had chosen landmark images because they had not wanted plan images. Their first step is to extract dense features from the image, than they select some features according to the algorithm they develop. They used Convolutional Neural Networks in the learning phase. They created a feature map depending on the characteristics and semantics of data. They had reduced the dimensions of the chosen features to improve the retrieval accuracy. Finally, they create an index tree and, they show the result.

There is a study which specifically focuses on a hashing method [5]. They propose this hashing method to be used as an image retrieval system on large scale datasets. They had also used Convolutional Neural Networks by developing this hashing method. They use image pairs to train the system. Their proposal only contains indexing. They index the query images by encoding them into binary code representations. Their main goal is to classify similarly encoded images and compute the binary encodings efficiently.

We see another recent research which focuses on image retrieval with the help of deep learning techniques [4]. Their method first creates a fixed length representation for each image. After that, using deep learning technique, they build features of the image regions by changing weights in each iteration. After the iterations are completed, they decide the final image descriptor. By this way, they clean the noisy image data. Thus, their accuracy result is better than other approaches' results. I think this approach is the least relevant approach to my method, because we do not care about noise for GIF image formats.

Most of the studies used state-of-the-art techniques with the extra steps of indexing and learning. However, now we will see a method [6] which tries to tune the widely known Convolutional Neural Networks technique. They create three dimensional models to improve the training data selection phase. They showed that extreme examples are very helpful to improve the retrieval performance. As image descriptors, they create three dimensional arrays, and they use one dimension, which corresponds to the feature maps, to represent the activations of the image. And, they used other two dimensional arrays to extract the feature vectors. So, they made some structural changes in the Neural Networks. They keep processing the learning steps by labeling and weighting the feature vectors. Finally, they reduce dimensions of features and use Bag of Words model to cluster the images. So, the data being unordered is not really important in this method.

As we can see in the recent works, nearly all researchers use machine learning techniques. I also used some of the machine learning techniques. For example, one of them is about calculating standard deviation which is closely related to the Gaussian Naive Bayes. And another example is about weighting the image features which is also closely related to the Neural Networks.

3 Methodology

When I started to make a research about this topic, my first goal was to determine the method to store GIF images, I needed to know how to store GIF images before how to fetch them. Most of the recent research shows that GIF formatted images are stored just like the other formatted images. There is no difference between image formats. Generally, the researchers propose and use a common method: they store images in file systems. Some database management systems have special features to store images in database. But researches had shown that storing images in a simple file system shows nearly the same performance comparing to the databases. A file system is easier to maintain. If the number of images in a directory increase exponentially, we can rearrange the folder, but in a database we don't have this option. Sometimes we are forced to rearrange because keeping too many images in one folder slows down the file access. Of course, storing images in the file system creates a disadvantage which is disk corruption or accidental deletion. We can handle this problem by using a source control system. By this way, we can retrieve any lost or deleted files. If we think about the security issues, the operating system can be as secure as databases. Storing images in a database comes with additional maintenance steps such as table configurations or software code reviews. Also, images can be easily cached when stored on the file system, because caching byte chunks is more costly. So, after the considerations of all these research, I decided to store GIF images in a file system folder.

As a general explanation, my proposed method contains four definitive steps. First one is to describe images. I use RGB color levels to describe images. Second step is to index the dataset. I create a hash map data structure to index the GIF images. But I create this hashmap after I made the compression calculations on the image. I find similarity result by doing these calculations. This corresponds to the third step.

As the final and fourth step, I perform some kind of searching operation in the folder. I scan the folder that contains GIF images, and I make calculations on all images until I find the matching GIF image. If I do not find any match, I set the result as false.

As a detailed explanation to the algorithm, my algorithm works sort of a compression algorithm but I use a feature of the image as descriptor. This algorithm works on the main memory. First, I split the images inside the GIF image and keep them in the memory. After that, I obtain red (R), green (G) and blue (B) values as double sized int arrays from each image, and I also keep them in the main memory. I calculate each row's standard deviations and as a result, I get three one dimensional arrays. I calculate means of all three arrays and I give weight to those arrays like this: the lowest mean valued array gets 0.5, the highest mean valued array gets 0.125, and the last array gets 0.25 weight. I gave these values according to the column's means because I think the standard deviation is high, the effect of resemblance between color levels is high. I multiple the weights with the matrices that I calculated by taking the standard deviations of color levels. Finally, I get a result matrix and it in a hashmap as value. The key of this hashmap is the image name. You can see the algorithm in Algorithm 1.

I do these algorithm steps on each image in the previously saved GIF images folder. I check if there is an equality between images, and I send true or false result to the end user. Calculating the difference between two images does not give satisfactory results according to the recent research. I generated a true or false result by looking for the given GIF image in my search folder. My algorithm's purpose is to give exact match result, not calculate the resemblance or correlation between images. My algorithm calculates the similarity between two GIF images by extending the Euclidean distance technique which can be seen in the equation (1).

By calculating the standard deviations and giving weights, I succeeded to find the similar images in a compressed dimension. If I keep the column that I find by calculating the standard deviation as an image feature and use it as an image descriptor, I could have improved the reusability, I could have take advantage of the revisitation process. I could have given relevance feedback for query reformulation if the given image to the system is not a GIF image. Those would have been advanced solutions.

The disadvantage of this method is that I do not use pixel selection as feature selection to reduce the overfitting. I also do not check the dense pixels, frequency of occurrence and partial matching that might change the matrices of calculations. However, this system has respectable advantages such that its scalability level is fine, and its performance is good on a huge dataset.

Algorithm 1 Indexing Algorithm

Require: Image G, input GIF image
1: Split GIF image into X number of images
2: Initialize image name imgName
3: **for each** image X in GIF image **do**
4: **for each** color level CL in an image **do**
5: Initialize mRGB[3] as color level matrices array
6: Initialize sdRGB[3] as standard deviation matrices array
7: Calculate CLM as color level matrix
8: **for each** row in CLM **do**
9: Add row to the corresponding matrix mRGB
10: **end for**
11: **for each** column in mRGB **do**
12: Calculate standard deviation matrix for color level
13: **end for**
14: Calculate mean of stddev matrix of color level
15: **end for**
16: Define hStdDevMat as the highest mean valued of color level
17: Define lStdDevMat as the lowest mean valued of color level
18: Define mStdDevMat as the middle mean valued of color level
19: Define descMat as descriptor matrix
20: Calculate descMat = 0.5 * hStdDevMat + 0.25 * lStdDevMat + 0.125 * mStdDevMat
21: Put key value pair <imgName, descMat> into the HashMap
22: **end for**

There is not any particular performance metric for images according to my research. In order to find the similar image in a folder, I have to calculate the difference between given image and previously saved image. So, this calculation must be done on each image in the folder. This might take some time but my proposed method is actually an algorithm to work on reduced and compressed image bytes. And, this will have a positive effect on the overall calculation time. I will stop when I find the exact matched image. I ignore the possibility of the same image with a different name, because I only care about the existence of given image.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{1}$$

4 Discussion and Conclusion

The proposed system has advantages and disadvantages. The most important disadvantage is that there are some extra work needs to be done until the indexing part. However, this comes with the advantage that you can retrieve images in a

more quickly and efficient time after you index the images. This will show even better performance when the dataset size is bigger. This technique is improvable with parallel processing techniques because there are several layers in the images. Those layers may easily be parallelized to the cores of processors or graphics cards. Since this method works like a compression algorithm, this method may be useful for higher sizes GIF images, because normally, GIF images are low sized but low resolution. With the help of this method, GIF images' sizes may be set as high. Furthermore, this method is not memory efficient but it is time efficient when it is processed on a big set of GIF images. My proposed solution might be used for other image formats. This solution might work faster for other formats, but the image to work on must have layers to process the method's steps. Also the storage method is file system, but another storage methods such as storing GIF images in a database might be proposed for improvement.

I presented an image retrieval system works only for GIF image formats. The beneficiaries of this system would be Web developers who works intensively with the images, because GIF images are mostly used on the Web. Image content providers and graphic card manufacturers might also be beneficiaries. The people who works on compression and hashing algorithms also might use this system to propose better algorithms. This area is still open for improvements.

References

[1] Paiz Reyes, Evelyn & Nunes, Nadile & Yildirim Yayilgan, Sule. (2018). GIF Image Retrieval in Cloud Computing Environment. 10.1007/978-3-319-93000-8_30.

[2] Jing Li, Bao-Liang Lu. (2009). An adaptive image Euclidean distance. Pattern Recognition, Volume 42, Issue 3, Pages 349-357, ISSN 0031-3203.

[3] Noh, Hyeonwoo and Araujo, Andre and Sim, Jack and Weyand, Tobias and Han, Bohyung. (2017). Large-Scale Image Retrieval With Attentive Deep Local Features. The IEEE International Conference on Computer Vision (ICCV), Pages 3456-3465

[4] Gordo, A., Almazán, J., Revaud, J., & Larlus, D. (2016, October). Deep image retrieval: Learning global representations for image search. In European conference on computer vision (pp. 241-257). Springer, Cham.

[5] Liu, H., Wang, R., Shan, S., & Chen, X. (2016). Deep supervised hashing for fast image retrieval. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2064-2072).

[6] Radenović, F., Tolias, G., & Chum, O. (2016, October). CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In European conference on computer vision (pp. 3-20). Springer, Cham.