A New Image Descriptor for Image Retrieval

Chenyang Zhao\(^1\), A. Ardeshir Goshtasby\(^1\), Shaodan Zhai\(^1\)
\(^1\)Computer Science and Engineering, Wright State University, Dayton, OH, USA

Abstract—Development of various image descriptors has greatly contributed to the advancement in image retrieval. Image descriptors can be made invariant to various image changes. In this paper, a new image descriptor is described and its application in image retrieval is evaluated. The descriptor uses the autoencoder concept to reduce the dimension of the feature vector measuring various properties of a keypoint. The new descriptor is found to have higher precision and recall rates when compared to the SIFT descriptor in image retrieval. Moreover, the descriptor is about three times faster than the SIFT descriptor, and by using the codebook concept, it can be made even faster.

Keywords: Image retrieval, Image descriptors, Autoencoder, Codebook representation

1. Introduction

With advancements in digital imaging technology, content-based image retrieval has received much attention during recent years. Image retrieval is the process of finding images in a database that are similar to a query image. Image retrieval methods can be categorized into Text-Based Image Retrieval (TBIR) and Content Based Image Retrieval (CBIR). Text-based image retrieval can be traced back to the late 1970s [1].

In TBIR, images are manually annotated and then searched by text. If images are annotated correctly, search results can be quite accurate; however, TBIR has some limitations. First, the amount of labor required to manually annotate all images in a database can be tremendous. Second, the inaccuracy caused by the subjectivity of human perception when generating the descriptions can be a source of error in image retrieval. Different individuals can have different interpretations of an image.

To overcome these limitations and drawbacks of TBIR, content-based image retrieval (CBIR) has been suggested. CBIR uses an images content to search for similar images in a database [2]. An image in a database is represented by a feature vector. The size of the feature vector is usually much smaller than the size of the image it is representing. The feature vector is used to compare and find similar images in a database.

Content-based image retrieval plays an important role in multimedia database systems. In CBIR, low-level features (such as colors, textures, and shapes) are used to describe an images contents. However, these low-level features can hardly describe the semantic concepts of an image [3]. In recent years, the mid-level features, such as SIFT descriptors, have attracted much attentions in image retrieval.

A feature measures a property of an image, either globally or locally [4]. Image features determine the performance of an image retrieval system; therefore, use of appropriate features is important in the success of CBIR. Image features such as color, shape, and texture can be used to match a query image to other images. Features are extracted automatically using computer vision techniques, and the similarity of the features is determined with those of features in images in the database [5]. In order to improve the retrieval performance, various combinations of features have been proposed.

In this paper, a new compressed image descriptor is proposed. Keypoints are extracted in the query image and in images in the database using the difference of Gaussian (DoG) operator, a descriptor is generated for each keypoint in the query image and in images in the database, the stacked autoencoder method is used to reduce the size of the image descriptors, and finally, the codebook method is used to speed up the retrieval process.

The remainder of this paper is organized as follows. Section 2 provides the background information about image retrieval, Section 3 lays out details of the proposed descriptor and provides information about the retrieval procedure and the codebook idea, Section 4 presents the results obtained by the proposed descriptor in image retrieval, and finally, Section 5 discusses strengths and weaknesses for the proposed descriptor.

2. Background

When a query image is provided, CBIR requires searching the database for the relevant images by using the contents of the images rather than relying on human-input metadata (such as captions or keywords). Image features and feature matching are important in CBIR. The performance of an image retrieval system depends on two factors: a suitable feature descriptor and a powerful feature matching strategy [6].

2.1 Keypoints detection

The first step in image retrieval is to detect keypoints in images. Various keypoint detectors have been proposed. In this paper, we use the DoG detector. The detector has three steps: (1) Scale-space extrema detection; (2) Keypoint localization; (3) Orientation assignment.
In the first step, keypoints are searched in scale-space by using the DoG operator. In order to detect extrema points from a stack of DOG images a point in the scale-space that is locally maximum within a $3 \times 3 \times 3$ neighborhood is taken to represent a keypoint. In the second step, a keypoint is determined by fitting a 3-D quadratic function using a second order Taylor expansion with the origin at the point of interest. Then, local extrema that correspond to weak edges are discarded. In the third step, a dominant orientation is assigned to each keypoint using the histogram of gradient directions in the neighborhood of the keypoint.

2.2 Creating a compressed image descriptor using a stacked autoencoder

An autoencoder is a specific form of an artificial neural network [7]. It is an unsupervised learning method that sets the target values to the input values. An autoencoder is composed of input layers, hidden layers, and output layers. A backpropagation algorithm is usually applied to train an autoencoder by propagating the error information (gradients) from the output layers back to the input layers [8] by using a gradient descent approach.

For an autoencoder with multiple hidden layers, it is difficult to optimize the weights by the back propagation algorithm. First, the back propagation algorithm does not work well in a deep neural network structure due to the diffusion of the gradients. Second, the optimization may get stuck at a false local minimum when the neural network is initialized with random weights. To alleviate this problem, Hilton [7] proposed a stacked autoencoder approach.

The motivation behind this is, gradient descent tends to work well when the initial weights are close to a good solution. To find initial weights for this good solution, a layer-wise pre-training process is used. This specifically consists of the following steps:

1) Train the bottom-most autoencoder, which consists of the input layer and the bottom-most hidden layer.
2) Remove the decoder layer of the trained autoencoder; and then construct a new autoencoder by taking the hidden layer of the previous autoencoder as input.
3) Train the new autoencoder.
4) Repeat Step 2–3 until all weights are pre-trained.

After the pre-training process, the next stage is to fine tune the weights of the network in a supervised fashion using the back propagation algorithm. In this stage, the weights obtained from pre-training are assigned as the initial weights, and the goal is to minimize the reconstruction error.

After detecting the keypoints, a local gradient patch centered at each keypoint is selected. Then, the image descriptor for the gradient patch is generated. For offline training, 40000 $41 \times 41$ patches were collected from diverse images. For each detected keypoint, the gradient patch centered at the keypoint is extracted. Then, the patch is rotated so its main orientation aligns with the x-axis, and scaled accordingly to the keypoints scale. For an image patch with size $39 \times 39$, the feature vector has 3042-values ($39 \times 39 \times 2$). A 3042-512-512-256-256-128-36 stack autoencoder is trained to extract feature vectors from the image. The feature vector can be computed by the local gradient image of the patch. We use the stacked autoencoder to project the feature vector from 3042 to 36. So our compressed autoencoder descriptor (AED) is significantly smaller than the standard SIFT descriptor with 128 values.

3. The retrieval process

The image retrieval process consists of two steps: the offline step and the online step. The offline step can be performed once ahead of time, and updated as new images arrive periodically. Thus, the efficiency of the offline step is not crucial. Since the online step will be performed when a query image arrives, this step should be carried out very efficiently. Figure 1 illustrates the image retrieval task. In the following, details of the offline step and the online step are provided.

3.1 Offline step

Given a database containing a large number of images, the offline step proceeds as follows:

- Using the SIFT detector extract keypoints in all images in the database.
- For each image, extract the local gradient patch centered at each keypoint and generate the gradient vector for each keypoint.
- Project each gradient vector to a compressed feature vector using the pre-trained autoencoder.
- Store the compressed feature vectors with the images.

3.2 Online step

In the online step, given a query image it is required to find all relevant images in the database. This is achieved as follows:

- Using the SIFT detector extract keypoints in the query image.
- Extract the local gradient patch centered at each keypoint, and generate the gradient vector for each keypoint.
- Project each gradient vector to a compressed feature vector by using the pre-trained autoencoder.
- For each image in the database:
  - Compute the similarity between this image and the query image. In the two images, compare each feature vector in one image with all feature vectors in the other image. If the Euclidean distance between two feature vectors is smaller than a required threshold value, declare a match. The similarity of the two images is calculated by the number of
matches. If the number of matches is greater than 0, return the image.

- Sort the retrieved images in descending order of their similarities with the query image.
- Display the relevant images.

In this image retrieval process, the bottleneck in computations is in the computational of the similarity between the query image and all the images in the database. By employing AED, the relevant images are obtained more efficiently than by SIFT.

3.3 Codebook image representation

Consider the matching process shown in Figure 1. Each image in the search database is compared with the query image. This is intractable in practice because a search database often contains a very large number of images. In order to speed up the matching process, a codebook mapping method [9], [10] is employed. The main idea is to map each image into a fixed-length vector, so that the similarity of two images can be efficiently determined by two vectors. Matching is then performed only on those images that produce the highest similarity scores.

Figure 2 shows the image retrieval process using the codebook mapping approach. In order to map an image to a fixed-length vector, we first cluster the (36-dimensional) AED feature vector of each keypoint for all images in the search database. We employ k-means clustering and set the number of clusters to $L$.

Each AED feature vector can then be assigned to one of the $L$ clusters. In this manner, an image can be described by a frequency distribution of these $L$ labels. This $L$-dimensional vector is called codebook or bag-of-words. This notion comes from the natural language processing area, and is a popular way of representing a document by its word frequency distribution, ignoring orders.

To employ this codebook mapping, there is one more step in the image retrieval’s offline step. After AED vectors of all images in the search database are obtained and stored, k-means clustering is performed on the AED feature vectors and group them into $L$ clusters. Then, these $L$-dimensional codebook vectors and the clustering model are stored.

In the new image retrieval system, the online step is updated as follows:

- Using the SIFT detector extract keypoints of the query image.
- Extract the local gradient patch centered at each keypoint, and generate a gradient vector for each keypoint.
- Project each gradient patch to an AED vector using the pretrained autoencoder.
- Map the query image to an $L$-dimensional codebook representation. More specifically, compute the distance between each AED vector in the query image and the $L$ cluster centers generated by the k-means algorithm in the offline step. Then, assign each AED vector to the cluster with its center closest to the AED vector.
- Compute the similarity between the query image and all the images in the database using their codebook representations. Select $N$ images with the highest similarity scores as the initially matched images.
- For each initial matched image in the search database:
Compute the similarity between this image and the query image. For two images, we compare each feature vector in one image with all feature vectors in the other image. If the Euclidean distance between two feature vectors is smaller than the chosen threshold, a match is declared. The similarity of the two images is calculated by the number of matches. If the number of matches is greater than 0, the image is returned.

- Sort the retrieved images in descending order of their similarity with the query image.
- Display the relevant images.

### 4. Evaluation Metrics

In the experiments, we use the popular metrics Precision vs. Recall to evaluate the performance of an image retrieval system. For any query image, Precision is the ratio of the number of relevant images retrieved to the total number of images (including relevant and irrelevant) retrieved:

\[
\text{Precision} = \frac{\text{Number of relevant images retrieved}}{\text{Total number of images retrieved}}
\]

where an image is considered relevant if and only if it is in the same group as the query image. Intuitively, Precision measures the quality of the retrieved images. The larger its value, the better the quality of the retrieved images. For simplicity, Precision of a query image is set to 1 if no images are retrieved.

Recall is the ratio of the number of relevant images retrieved to the total number of relevant images retrieved in the entire search database:

\[
\text{Recall} = \frac{\text{Number of relevant images retrieved}}{\text{Total number of relevant images}}
\]

In some real-world applications, the total number of relevant images is usually unknown. For example, if the search space is the entire Internet, it is hard to know how many images are relevant to a query images. However, in our application, Recall is computable as we assume that only if images in the same group are relevant. Recall measures the ability to find relevant images by the image retrieval system.

With the Precision and Recall for a query image, we can compute the average Precision and Recall over all the queries. Ideally, it is desired that an image retrieval system provide both a high Precision and a high Recall rate, but this is often an unattainable goal in practice. For the extreme cases, if no images are retrieved, the Precision is set to 1, but obviously this image retrieval system is useless. On the other hand, if an image retrieval system retrieve all the images in the database, then Recall is always 1, but Precision can be close to 0. Therefore, there is usually a trade-off between precision and recall, and we employ the Precision vs. Recall curves to measure the performance of an image retrieval system. The closer the curve is to the top of the chart it indicates a better performance.

### 5. Results

In order to evaluate the performances of the proposed descriptor, the INRIA Holidays dataset and ORL database are used. The experiment is to compare the retrieval performance between our descriptor and the SIFT and PCA-SIFT [11] descriptors. For the image retrieval systems that use SIFT and PCA-SIFT, they are very similar to the process of Figure 1. The only difference is replacing the AED module to SIFT or PCA-SIFT.

For a given dataset, we first divide a dataset into two parts. The first part contains all the query images, and the rest is used to search the database. For a query image, the goal is to try to find all relevant images for the query. Taking the ORL dataset as an example, there are 40 query images in total, and the remaining 360 images represent the database. For a query image, we intend to find the other 9 images of the same person from the entire 360 images.

#### 5.1 Image retrieval results using the Holidays dataset

The Holidays dataset\(^2\) was created for the ANR RAFFUT project. It contains 1491 personal vacation photos with a very large variety of scene types, including natural, food, water and building. There are totally 500 image groups in this dataset, each of them represents a distinct scene of object. For each image group there are 2 to 10 images, and these images were taken from different viewpoints or by varying the lighting. We choose the first image of each group as the query, and 1 to 9 images of the group as the correct retrieval results.

Figures 3 and 4 illustrate the retrieved images of SIFT, PCA-SIFT and AED algorithms. For figures 3 and 4, AED descriptor performs especially well, finding all retrieved images.

The Precision vs. Recall curves of SIFT, PCA-SIFT, and AED are shown in Figure 5. For each query image, each algorithm is allowed to return at most 9 images. This result shows that AED is comparable or better than SIFT and PCA-SIFT in image retrieval when using the Holidays dataset. We believe this is due to AED’s high matching accuracy at the keypoint level, which also translates to high retrieval results.

Since the codebook mapping is an approximate method, the matching performance may worsen when compared to the naive matching approach. Figure 6 shows the Precision vs. Recall curves of AED and AED with codebook mapping. After initial matching with codebook representation, some relevant images are dropped incorrectly. However, this loss of accuracy may be acceptable when considering improvement run time.

\(^2\)https://lear.inrialpes.fr/ jegou/data.php
query image:

retrieved images by SIFT:

retrieved images by PCA-SIFT:

retrieved images by AED:

Fig. 3: Retrieved images by SIFT, PCA-SIFT, and AED for the query image 136000.pgm.

5.2 Image retrieval results using ORL Dataset of Faces

The ORL database was created by AT&T Laboratories^3. It contains a set of face images of 40 individuals taken between April 1992 and April 1994 at the lab. For each person, there are 10 different face images, and each image contains just one face. The 10 face images for a person are different due to four factors: 1) images were taken at different times; 2) varying the lighting; 3) facial expressions, such as open/closed eyes, smiling/not smiling; 4) facial details, such as glasses/no glasses. All the images have black homogeneous background. The size of each image is 92 × 112 pixels, with 256 grey levels per pixel. We divide the face images into 40 groups, each group contains 10 images for one person. For each group, we choose the first image as the query image, and thus there are 9 relevant images for any query image.

From the experimental results using the ORL dataset, the Precision vs. Recall curves of SIFT, PCA-SIFT, and AED are plotted in Figure 9. As before, at most 9 images are retrieved for each query image. The image retrieval task on

query image:

retrieved images by SIFT:

retrieved images by PCA-SIFT:

retrieved images by AED:

Fig. 4: Retrieved images by SIFT, PCA-SIFT, and AED for the query image 132500.pgm.

Fig. 5: Precision vs. Recall curves of SIFT, PCA-SIFT, and AED using the Holidays dataset.

^3http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
Fig. 6: Precision vs. Recall curves of AED and AED with codebook mapping when using the Holidays dataset.

query image:

retrieved images by SIFT:

retrieved images by PCA-SIFT:

retrieved images by AED:

Fig. 7: Retrieved images by SIFT, PCA-SIFT, and AED for query image of the 7th and 2nd persons.

this dataset is easier than the previous Holidays dataset. The retrieved results are perfect for each algorithm if we only consider the first retrieved image. Considering the overall retrieval results, AED performs better than SIFT and PCA-SIFT on this dataset.

Figure 8 shows the retrieved images for the query image of the 8th person and the query image of the 3rd person. The image retrieval tasks for these two query images are more difficult, and each algorithm retrieves some irrelevant results. For the query image of the 8th person, 7 relevant images are retrieved by all the three algorithms. However, PCA-SIFT performs worse than the other two methods since the irrelevant images rank higher. For the query image of the 3rd person, 6 relevant images are retrieved for both SIFT and PCA-SIFT. AED performs better for this query image since 8 relevant images are retrieved.

5.3 Run-Time Analysis

Besides the accuracy, an important performance measure of an image retrieval system is its speed. As was discussed before, the matching process is the bottleneck in image retrieval since it has to be executed online. To evaluate the time performance of the image retrieval time, each method was tested on a Red Hat Linux server with Intel Xeon with 5650 CPUs. The algorithm was implemented in C++. Here an example using two images from the Holidays dataset.

Fig. 8: Retrieved images by SIFT, PCA-SIFT, and AED for the query image of the 8th and the 3rd persons.

Fig. 9: Precision vs. Recall curves of SIFT, PCA-SIFT, and AED when using the ORL dataset.

Fig. 9: Precision vs. Recall curves of SIFT, PCA-SIFT, and AED when using the ORL dataset.
is given. Overall, 2967 and 3284 keypoints were detected by the SIFT detector, requiring about 9.7 million point comparisons during the matching process. Table 1 shows the run times in ms for SIFT, PCA-SIFT, and AED. PCA-SIFT and AED spent almost the same time in matching as the feature vector in each keypoint had 36 dimensions by both methods. SIFT is about 3 times slower than PCA-SIFT and AED. This is due to associating a 128-dimensional feature vector to each keypoint. For each query image, there was a need to use all 1491 images in Holidays dataset, requiring about one hour matching time for AED and PCA-SIFT, and three hours for SIFT. Obviously this run time is not practical in real applications.

When we use the codebook method, and set $N$ to 100 in our image retrieval system, the query image is compared to 100 images instead of all images in search database. For the Holidays dataset, this speeded up the process by about 15 times.

### 6. Conclusions

With development of various image descriptors, significant progress has been made in image retrieval. Image features generated by local image descriptors are generally invariant to different image transformations. This paper described and evaluated a new image descriptor using the autoencoder concept.

The stack autoencoder is used to reduce the dimension of the feature vectors describing the properties of the neighborhood of a keypoint. Compared to the SIFT descriptor, which produces a 128-dimensional feature vector, the proposed encoder produces a 36-dimensional feature vector. As a result, the proposed descriptor has a considerably lower computational requirement than the SIFT descriptor.

When used in image retrieval, the proposed descriptor is found to have a higher combined precision and recall rate than SIFT. Moreover, the proposed descriptor is about three times faster than the SIFT descriptor. By using the codebook method, the speed of the proposed descriptor in image retrieval is further increased.

### References


