Rapid Face Image Retrieval for Large Scale Based on Spark and Machine Learning

Yuting Zeng*, Yujin Li, Qiming Hu and Yuan Tang*

School of Information Science and Technology, Chengdu University of Technology, Chengdu 610059, China
Email: 949261626@qq.com; tangyuanmail@gmail.com

Abstract. This paper proposed a distributed computing environment large-scale face image retrieval method. Based on Spark platform, a face retrieval method based on machine learning is established, which takes advantage of parallel computing to improve the efficiency of face image retrieval. In the method, SIFT algorithm is used for feature coding, PCA dimension reduction processing, HBase database is used for data storage, and KD-Tree query algorithm is used to match images similar to the query images. Meanwhile, large-scale computing engine uses Spark to process data to improve the retrieval efficiency. The CelebA dataset is selected to test the method, and the experimental results show the effectiveness of the method.

Keyword. Image retrieval; Spark; parallel computing; machine learning.

1. Introduction

Development of image retrieval technology is a process from quantitative change to qualitative change and then from qualitative change to quantitative change. In the 1970s, image retrieval at that time was still text-based (TBIR) [1], which required manual annotation of images. In the 1990s, content-based image retrieval technology (CBIR) [1] emerged, which can simulate human vision to analyse and process images from the perspective of computers, identify images with features such as colour, texture and shape, and use similarity between features to retrieve images. As human officially entered the age of big data, image data is transformed from small data volume to large data volume or even large data volume, in the age of big data, as a branch of image retrieval, face image retrieval is faced with many technical difficulties, including how to reduce the calculation and how to improve the efficiency of image retrieval [2]. Traditional data processing methods cannot meet the requirements of big data due to the long processing time, so we need to use distributed storage and computing technology based on big data. The combination of big data storage and computing technology with traditional image retrieval technology makes it feasible to retrieve large-scale face images quickly or even in real time [3].

In the aspect of image retrieval [4], Gudivada and others established a retrieval system based on image features. Hyun-Chul Kim adopted a one-dimensional and two-dimensional hybrid method to achieve database index; Eickeler used a pseudo-second-order hidden Markov model to retrieve images from an image database; Guoxia Sun combined PCA and ICA methods to produce more easily distinguishable content features and index images.

Based on big data computing engine Spark and open source computer vision library OpenCV, a distributed fast image retrieval system is designed and implemented. Using OpenCV built-in cascade classifier (based on Viola & Jones algorithm) to detect each image in the image dataset, and get the
local image of the image. SIFT feature extraction algorithm is used to extract the feature values of local images, and PCA dimension reduction algorithm is used to reduce the dimensions of the obtained high-dimensional features. Then the image characteristic values obtained by dimensionality reduction are stored in the HBase database. KD-Tree is used to build the index, and the algorithm is applied to the fast matching of image features.

Each chapter is arranged as follows: Chapter 2 introduces the relevant technologies of face retrieval methods in this paper. Chapter 3 mainly introduces the face retrieval architecture and the method of Spark computing framework to realize distributed image retrieval. Chapter 4 is the test and result analysis of the method in this paper. Finally, the conclusion and future work.

2. Methods

2.1. Viola&Jones Image Detection
Viola&Jones (Hereinafter referred to as VJ) image detection is a classical algorithm. The core of VJ detector mainly consists of three parts:

(1) The eigenvalue of the image is computed quickly by the integral channel.
(2) Screening features (AdaBoost classifier).
(3) A cascade classifier is used instead of Adaboost to quickly discard non-image features.

Haar-like sums the values of white pixels in the figure to have Haar-like characteristics, and then makes difference calculation with black pixels. There is often more than one such feature point in the image, so integral graph is used. The difference in the integrogram is that the pixel value at each point is the sum of all the pixels in the upper left corner. The expression is shown below:

\[
s(x, y) = \sum_{x' \leq x} \sum_{y' \leq y} f(x', y')
\] (1)

In this equation, \((x, y)\) represents any point on the integral image, \(s(x, y)\) represents any point on the integral image, \((x', y')\) represents any point on the upper left corner of point \((x, y)\), and \(f(x', y')\) represents the pixel value of point \((x', y')\).

The integral image satisfies the following formula:

\[
s(x, y) = f(x, y) + s(x-1, y) + s(x, y-1) - s(x-1, y-1)
\] (2)

In order to select appropriate features from many features to differentiate images, the Adboost method can be used.

2.2. Feature Extraction
SIFT (Scale Invariant Feature Transformation) is a special stable local feature [5]. The algorithm implementation mainly has the following steps:

2.2.1. Generating Scale Space. Keeping the image resolution unchanged, different parameters are used to blur the image to get the Gaussian scale space. Different Gaussian kernels are convolved with the image to obtain the Gaussian scale space of the image:

\[
L(x, y, \sigma) = G(x, y, \sigma) \times I(x, y)
\] (3)

\[
G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}
\] (4)

In the expression, \(G(x, y, \sigma)\) represents the Gaussian kernel function, \((x, y)\) represents the spatial coordinate, \(\sigma\) represents the scale space factor, and the larger the value is, the more blurred the image will be. \(L(x, y, \sigma)\) said image gaussian scale space.
Next, feature points can be detected, usually using difference of Gaussians. Set $k$ for two adjacent Gaussian scale space scale factor, and the definition of the difference of Gaussians is shown as follows:

$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] \ast I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

(5)

2.2.2. Detect the Extreme Point. Comparing adjacent at each sampling point [6], and the intermediate detection point is compared with the points in the adjacent eight directions and 18 points at the upper and lower adjacent scales to ensure that the extremum points can be detected in both spaces. The feature points are the largest or smallest in the 26 domains of this layer and the upper and lower layers.

2.2.3. Specify the Direction of the Feature Points. The direction of each key point is assigned, and the direction, position and scale of the key point are transformed to determine the SIFT feature area.

2.2.4. Descriptive Feature Points. The pixel gradient in the $16 \times 16$ window range around the feature point is calculated, and the weight away from the centre is reduced by Gaussian descent method. An $4 \times 4 \times 8 = 128$ dimensional descriptor is shaped for every feature point. The extracted high-dimensional features lead to large resource cost in the process of feature storage and feature matching in the later stage, so it is necessary to process the extracted high-dimensional features, namely feature dimensionality reduction [7]. PCA (principal component analysis) is a commonly used feature extraction method. By transforming the data into linear independent representation, the high-dimensional data is mapped to a low-latitude space, thus reducing the data dimension while retaining the main features [8, 9].

2.3. KD-Tree

KD-Tree is a kind of binary tree, which stores K-dimensional data (K>1). It is often used for neighbor search in large scale high dimensional data space [10].

KD-Tree algorithm usually contains two parts:

• Establishment of K-D tree.
• Based on KD Tree index structure of the nearest neighbor search.

The construction process of KD-Tree is shown in table 1.

| Process | Specific operation |
|---------|--------------------|
| 1       | Find the dimension with the largest variance, select the middle value $M$ as the center to split the dataset, get two subsets, and create a tree Node for storage. Process 1 is repeated for both subsets until all subsets can no longer be partitioned. When reach cannot be further divided into subsets, belongs to a subset of the data stored in the leaf node. |
| 2       | After constructing KD-Tree, can be the nearest neighbor search. The nearest neighbor search process of KD-Tree as shown in table 2. |

Table 1. KD-Tree construction process.

| Process | Specific operation |
|---------|--------------------|
| 1       | Starting from the root node, the value of the data $Q$ corresponding to the k-dimension of the node to be queried is compared with the median value $m$. Record the data points corresponding to the minimum distance, denoted as the current “nearest neighbor point” current Point and the minimum distance current Distance. |
| 2       | Trace back to find the “nearest neighbor point” closer to $Q$, and see if there is a point whose distance from $Q$ is less than current Distance. |

Table 2. The nearest neighbor search process of KD-Tree.
3. Large-Scale Face Image Retrieval Framework

3.1. Hadoop Cluster

Deploy the project to distributed environment, including Hadoop environment, HBase distributed database and Spark environment. This experiment uses the cloud server to complete the fully distributed cluster deployment.

The Hadoop cluster of this system contains one Master node and two Slave nodes. The detailed configuration is shown in table 3.

| Node name | Operator | System | CPU | RAM | Hard Disk |
|-----------|----------|--------|-----|-----|-----------|
| Master    | Baidu Cloud | CentOS 7.3 x86_64 | 2vCPUs 4GB | 40G |
| Slave1    | Hicloud   | CentOS 7.4 64bit | 2vCPUs 4GB | 40G |
| Slave2    | Tencent Cloud | CentOS 7.3 x86_64 | 2vCPUs 4GB | 50G |

The HBase cluster contains one HMaster(Master) node and one HRegionServer(Slave1) node, which still adopts the strategy of first configurable at Master and then distributing to Slave.

Spark relies on a Hadoop environment and has the Scala language installed and configured.

3.2. Spark Image Detection and Feature Extraction

(1) Firstly, Spark uses RDD to read image files from the original face image data set in the Linux file system, where the key is the image path and the value is the byte array of the image.

(2) For each face image, the following steps are performed based on OpenCV iteration:

- Image preprocessing. The size of the image is normalized, and the grayscale image corresponding to the original image is obtained by grayscale processing.
- The local image of the face image position is obtained by image detection on the grayscale image, is shown in figure 1.

- PCA-SIFT is used to extract features from facial images, and the corresponding eigenvalues are obtained. The process of extracting image key points is shown in figure 2: (a) shows the local image of the original image, and (b) shows the key points of SIFT detected according to the local image.

(3) Spark stores the original image, local image and extracted facial features into the HBase database. RowKey is the file name of the original image.

3.3. Construction of KD-Tree Index for Image Features

On the basis of feature extraction, the KD-tree algorithm in Spark and OpenCV is used to construct the KD-tree index, so as to accelerate the efficiency of retrieval. The process is as follows:
• Spark uses RDD to read the image feature values stored in the HBase database. Key is the rowKey corresponding to the image, and value is the feature value of the image corresponding to the rowKey.
  • Based on the read data, the KD-Tree index structure is established.
  • Save the created indexes to the index table of the HBase database.

![Image](a)(b)

**Figure 2.** Extraction of image key points.

### 3.4. Distributed Retrieval

The process of performing distributed image retrieval on Spark can be described as follows:

- The program first reads the contents of the index table and builds the index.
- Then the feature corresponding to the user input image processed by the feature extraction module is searched by using the established index. Finally, it returns the leaf node which is closest to the original image, and the approximate image is extracted from the leaf node information and returned to the user.

The pseudocode for the index lookup job is shown in figure 3.

```plaintext
Class KDSearch

Input: image features to be queried: queryDescriptors, index: indexdata
Output: return a list of searched images: matchList

1. function search(featureMat)
2. configure Spark: SparkConf context = SparkContext(sparkConf)
3. create index RDD: datafile < -indexdata
4. Set up the matcher using the FLANNBASED library provided by OpenCV
   descriptorMatcher = DescriptorMatcher(event(FLANNBASED))
5. List<MatOfDMatch> matchList = new ArrayList<>(); Storing query results
6. bnmMatch(queryDescriptors, indexdata, matchList, 15)
7. return matchList
8. end function

**Figure 3.** Distributed retrieval of pseudocode.

### 3.5. System Architecture

This system is based on Hadoop+HBase to store data. HDFS is adopted as the bottom layer, and HBase is used to store images and extracted image features on it. On top of the system is the Spark big data computing engine, which is responsible for the relevant data distributed processing. Meanwhile, Spark relies on OpenCV computer vision library to complete image detection, feature extraction and KD-Tree retrieval.

### 4. Experiment and Performance Evaluations

#### 4.1. Data Set

This system uses CelebA data set, which is an open source large-scale face image data set. It includes 10177 face images, and the completed data set contains 202599 face images. The face data in the data set are all from the network space, which is closer to the real situation, and the data amount is
moderate. At the same time, it contains features of eastern and western faces, so it is used in the face retrieval system designed in this paper.

4.2. Recall and Precision
When retrieving the face images corresponding to ID 2880, a total of 13 face images are retrieved. The first 10 images are the images corresponding to ID 2880, and the last three images are false positives. Thus, the precision rate: $\frac{10}{13} \times 100\% = 76.9\%$, and the recall rate: $\frac{10}{15} \times 100\% = 66.7\%$.

When retrieving the face images corresponding to ID 4407, a total of 15 face images are retrieved. The first 11 images are the images corresponding to ID 4407, and the last four images are false positives. Thus, the precision rate: $\frac{11}{15} \times 100\% = 73.3\%$, and the recall rate: $\frac{11}{15} \times 100\% = 73.3\%$.

When retrieving the face images corresponding to ID 3354, a total of 12 face images are retrieved. The first 10 images are the images corresponding to ID 2880, and the last two images are false positives. Thus, the precision rate: $\frac{10}{12} \times 100\% = 83.3\%$, and the recall rate: $\frac{10}{15} \times 100\% = 66.7\%$.

5. Conclusions and Future Prospects
Large-scale face image retrieval has strong practical value in real life. Here, we mainly study the impact of Spark computing engine on large-scale face image retrieval. We used Viola&Jones framework for face detection, SIFT algorithm for feature coding, PCA algorithm for dimension reduction, HBase database for data storage, and KD-tree query algorithm for matching. However, this work is still in its early stages and we will continue to investigate its applications in the future. At the same time, we will further use other large-scale data sets to train and optimize the model.

Acknowledgments
This work was sponsored by Science and Technology Program of Sichuan Province, China (Grant No. 2020YFS0090, 2021YFN0117).

References
[1] Wang M and Deng W H 2021 Deep face recognition: A survey Neurocomputing 429 215-244.
[2] Huang Y Y, Tang Y and Xiong T S 2020 Large-scale face image retrieval based on Hadoop and deep learning International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP) pp 326-329.
[3] Liang Z J and Zhao Y L 2020 Image classification based on high-dimensional feature retrieval intelligent algorithm Microprocessors and Microsystems 103497.
[4] Zheng H L 2015 The Research and Realization of Large-Scale Face Image Retrieval System (University of Electronic Science and Technology of China) Master’s Thesis.
[5] Pedro F, David M and Deva R 2008 A discriminatively trained, multiscale, deformable part model IEEE Conference on Computer Vision and Pattern Recognition pp 1-8.
[6] Ke N Y and Sukthankar R 2004 PCA-SIFT: A more distinctive representation for local image descriptors Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
[7] Dong Z, Wei J, Chen X and Zheng P 2020 Face detection in security monitoring based on artificial intelligence video retrieval technology IEEE Access (8) 63421-63433.
[8] Sachin S F, Mohammad S and Li L J 2015 Multi-view face detection using deep convolutional neural networks Proceedings of the 5th ACM on International Conference on Multimedia Retrieval.
[9] Wang C, Fan C and Wu Y 2019 Large-scale real-time face retrieval system based on Storm Computer Systems and Applications 93-98.
[10] Lowe D L 1999 Object recognition from local scale-invariant features Proceedings of the International Conference on Computer Vision.