**Image Feature Layered Indexing Method for Transparent Access**

Hong ZHANG\(^1\) and Yu-jie LIU\(^2\)

\(^1\)China Electric Power Research Institute (Nanjing), Nanjing 210003, Jiangsu Province, China
\(^2\)Zhejiang University, Yuquan Campus, Hangzhou, Zhejiang Province, China

**Keywords:** Data transparent, Image feature, Hierarchical indexing, Pyramid image index structure.

**Abstract.** In the data transparent access process, it is often necessary to query the data accurately to find the location of the data that the user needs to access. This paper studies the image feature organization and indexing of image data based on image content retrieval, and uses the image plane position constraint relationship to propose an image feature layered index method. In the process of image transparent access, data storage management needs to detect and extract local feature information of the image (such as SIFT feature points), and then design appropriate data structures to organize and index massive high-dimensional feature vectors, and in a specific storage structure. Design an efficient and accurate feature search algorithm. The paper mainly includes two parts: LSH high-dimensional indexing algorithm for thumbnails and KD-Tree high-dimensional indexing algorithm based on improved position information.

**Introduction**

This chapter mainly introduces a hierarchical indexing method for image features of large images. The method utilizes large image global position constraint information unique to large image problems to establish an image hierarchical index structure. This method combines the LSH algorithm with the KD-Tree algorithm based on location information improvement \([1, 2, 3]\). It can control the spatial complexity of the LSH algorithm and the time complexity of the KD-Tree algorithm in high-dimensional feature retrieval, speed up the search and search speed, and improve the correct rate of finding neighbor features, thus obtaining compared with the LSH algorithm and the best comprehensive search performance of the KD-Tree algorithm.

The method mainly includes extracting thumbnails to establish a two-layer pyramid image index structure, using the LSH algorithm for rough location search for thumbnails, and using the KD-Tree algorithm based on position information improvement for image original maps to perform feature point neighbor search three steps. As shown in Figure 1.

![Figure 1. Schematic diagram of hierarchical indexing method for large image features.](image)

The method first extracts a large image from a two-layer pyramid image. Then in the upper thumbnail, we use the LSH method for fast retrieval. Because the image is small at this time, it does not take up too much memory space, and the LSH is a neighbor search that can return a rough feature.
location search result. Finally, the coarse position information of the upper layer feedback is mapped to the auxiliary KD-Tree in the lower layer for searching. In the process of building the lower KD-Tree, we also add the X and Y position information to the feature vector, and split the nodes according to the X and Y position coordinates in the first four layers of the tree. Thereby, the number of feature points to be matched is reduced by introducing position information to overcome the weakness of the KD-Tree search speed being too slow when the retrieval feature is too much.

**LSH High-dimensional Index Algorithm for Thumbnails**

As shown in Figure 1, we can get the same scale thumbnail of the original image of the large image by downsampling the original image. For example, in the experiment designed in this chapter, the length and width of the thumbnails we get are 1/4 of the original image, that is, the size of the thumbnail is 1/16 of the original image size. By performing LSH on the thumbnails, the time complexity of the LSH algorithm can be reduced. For the LSH index method of thumbnails, this section will elaborate on the following two aspects: one is the establishment of the LSH index structure, and the other is the neighbor search method for the LSH to return coarse position information.

**Establishment of LSH Index Structure**

LSH (Location Sensitive Hash) local perceptual hash function, which is different from the general hash function is its position sensitivity. That is, the similarity before hashing can be similar to some extent after hashing, and has a certain probability guarantee. In [11], the LSH algorithm has been improved. The specific implementation of most LSH algorithms is based on this, and this paper is also based on this.

According to [4, 5, 6, 7], LSH can solve the approximate query problem under high-dimensional vectors, which is defined as:

For any q, p belongs to S, If the function set $H = \{h_1, h_2...h_n\}$ from the set S to U pairs the distance function $P'_h$, such as the Euclidean distance satisfies the conditional formula 1, the $p_h = Unary_c(p[0])...Unary_c(p[d-1])$ is said to be position sensitive:

$$
\text{If } D(p, q) \leq r, \text{ and } Pr (h(p) = h(q)) >= p1 \\
\text{If } D(p, q) > r(1+\varepsilon), \text{ and } Pr (h(p) = h(q) <= p2
$$

(1)

For this paper, the index created by the local perceptual hash function algorithm contains 10 hash tables, and each hash table $T_i(i = 0, ..., 10)$ contains 10 buckets for storing data. For hashing and compression purposes, we need two hash functions here. The first is a hash function set containing 10 concrete hash functions, $g_i(i = 0, ..., 10)$, the first is a ten hash table general hash function h. The local perceptual hash function maps vectors of similar distance to the same bucket in the sense of probability, so a rough set of matching points is returned.

According to [8, 9, 10, 11, 12, 13, 14], the steps for establishing an LSH index structure for thumbnails are as follows:

1) Convert the 128-dimensional vector value of each SIFT feature vector P extracted from the thumbnail into the binary vector $p_h$ in the Hamming space $H^{cd}$, that is, convert each feature vector into a binary sequence of 0 and 1.

$$
p_h = Unary_c(p[0])...Unary_c(p[d-1])
$$

(2)

$Unary_c(x)$ denotes that the total length of the binary vector is c, where x consecutive 1s exist. For the 128-dimensional SIFT feature, a binary sequence of $128 \times 255$ length is formed after the conversion (because the feature point has a value between 0 and 255 for each dimension after the
feature vector is normalized). The purpose of this processing is to make the distance in the Hamming space consistent with the distance in the 128-dimensional feature space of the original image.

2) Here, the first hash function set $g_i$ is applied to $p_h$ to make the first hash, as shown in Equation 3:

$$P'_h = g_i(p_h)$$  \hspace{1cm} (3)

The role of each $g_i$ in the set of hash functions is to randomly select the k-dimension on $p_h$ according to a specific rule, that is, randomly select k values in the binary sequence obtained in 1) according to a specific rule (in the specific operation process herein, k is 947 dimensions. This is determined by multiplying the length of the binary sequence by an value factor and composing a k-dimensional binary vector $P'_h$, where the formalized representation of $P'_h$ is

$$P'_h = \{idx_1, idx_2, ..., idx_k\}$$

3) For the one-dimensional vector $l_i = h(p'_h)$ composed of the new k-dimensional (947-dimensional) binary vector obtained in 2), we use the second hash function $h$ to obtain the final hash result $l_i$, as shown in Equation 4.4:

$$l_i = h(p'_h)$$  \hspace{1cm} (4)

Among them, the hash function $h$, we use the MD5 algorithm. MD5 (Message Digest Algorithm) Chinese name is the fifth version of the message digest algorithm, which is a widely used hash function to provide message integrity protection. MD5 compresses the large-capacity annotation information into a simple format, which is to convert a byte string of arbitrary length into a string of hexadecimal digits of a certain length.

A brief description of the MD5 algorithm can be: MD5 processes the input information in 512-bit packets, and each packet is divided into 16 32-bit sub-packets. After a series of processing, the output of the algorithm consists of four 32-bits. The grouping consists of cascading the four 32-bit packets to generate a 128-bit hash value.

4) Put the MD5-compressed one-dimensional vector $l_i$ obtained in 3) into the corresponding hash table $T'_i(i = 0, ..., 10)$ (so-called "bucket"). The hash table $T'_i(i = 0, ..., 10)$ is stored in a red-black tree for quick searching. At this point, the LSH index structure is constructed, and the original 128-dimensional feature value of each SIFT feature point is hashed into 10 hash tables. Each hash table stores a 128-bit binary sequence of 0 and 1.

Near-neighbor Search Method for Returning Coarse Position Information by LSH

Based on the establishment of the LSH index structure in the previous section, we can use the index structure to perform fast feature point neighbor search. The schematic diagram of the search flow is shown in Figure 2.
The search steps are as follows:

1) First, a set of SIFT feature points of a captured image of a mobile phone is extracted, and the following operations are performed for each of the feature points Q.

2) For the feature point Q, first convert to the binary vector in the Hamming space $H^{cxd}$ by the formula 2.

3) For $H^Q$, use the first hash function set $g_i$ to act on $H^Q$ to make the first hash, as shown in Equation 5:

$$Q'_H = g_i(Q_H)$$

Randomly select the 947 dimension on $H^Q$ according to a specific rule to form a 947-dimensional binary vector $Q'_H$, where the formalized representation of $Q'_H$ is $Q'_H = \{idx_1, idx_2, ..., idx_{947}\}$.

4) We use the second hash function $h$ to get the final hash result $l_i$, as shown in Equation 6, where h also uses the MD5 algorithm:

$$l_i = h(Q'_H)$$

5) The matching points $P_i$ corresponding to each $T_i$ are queried in ten hash tables $T_i(i = 0, ..., 10)$ of the LSH index structure. We collect the results of the ten hash tables and find the feature points that have the most occurrences among the ten matching results. Because in the specific experiment, we use LSH to perform 1-neighbor search, that is, nearest neighbor search, so here we return the most feature points in the ten hash tables. If a K-nearest neighbor lookup is performed, the top K feature points that have the most occurrences are returned.

So far, the LSH nearest neighbor search for the thumbnail is completed, we can use the position information of the nearest nearest neighbor matching feature point to assist the improved KD-Tree based neighbor search in the original image.
KD-Tree Neighbor Search Method for Retrieving Information by LSH

After the improved KD-Tree index structure based on location information is established, we can perform neighbor search based on the coarse location information returned by the upper layer thumbnail. The search procedure is as follows.

1) According to the coordinate \((x_0, y_0)\) of the X and Y of the nearest neighbor of the upper layer thumbnail returned by the LSH, we map it to the position coordinate \((x, y)\) of the lower layer according to the scale, as shown in Equation 7, where \(k\) is the original image and the thumbnail image.

\[
x = kx_0
y = ky_0
\] (7)

2) The feature points in the image captured by the user's mobile phone that are correctly matched in the LSH algorithm are filtered out, and the secondary matching based on the improved KD-Tree is performed. In the quadratic matching process, firstly, the nearest neighbor nodes returned in the LSH algorithm according to each feature point are searched in the first four layers of the improved KD-Tree by 1) mapping the position information in the original image. This allows you to locate a precise sub-area in the original image. For example, in the experiment of this paper, the sub-region of the original image \(1/16\) can be reduced by the retrieval of the feature points of the step, and the number of feature points that are theoretically retrieved is also reduced to about \(1/16\) of the initial feature point number.

3) After 2) positioning to the sub-region of the large image plane, the BBF-based neighbor search is performed on the KD-Tree according to the feature information of the 128-dimensional retrieval feature point.

BBF (Best Bin First) is an improved KD-Tree nearest neighbor query algorithm. The "backtracking" in the KD-Tree search process is determined by the "query path" and does not take into account some of the properties of the data points themselves on the query path. The BBF query idea is to sort the nodes on the "query path", such as sorting the distance between the hyperplane (called Bin) and the query point. Backtracking always starts with the tree node of the highest priority (Best Bin). In addition, BBF also sets a running timeout limit. When all nodes in the priority queue are checked or exceed the time limit, the algorithm returns the best result currently found as the approximate nearest neighbor. KD-Tree can be extended to high-dimensional datasets using the best-bin-first search method.

The BBF algorithm is implemented with a priority queue (usually with a minimum heap). Starting from the root, when performing feature search on KD-tree, first perform deep traversal to find the leaf node, and the temporarily missed node is first placed in the priority queue; then the current key value is the smallest from the queue (here refers to If the distance in the current splitter split dimension is the smallest, repeat the above process to traverse the leaf node process until the priority queue is empty, or the specific number of steps has been repeated.

After the above steps, the BBF neighbor search on the improved KD-Tree can be completed. In this paper, we return the 2-nearest information of the feature points.

Experimental Results and Analysis

According to the theoretical analysis, the indexing algorithm proposed in this paper adds another layer of LSH index for thumbnails on the KD-Tree index structure of the original image, and the thumbnail size is only \(1/16\) of the original image, so it does not occupy too much. Memory space. Moreover, since
the upper layer KD-Tree search and backtracking range can be reduced to the entire feature space point set by the auxiliary position information of the upper layer, the proposed method has higher retrieval speed and correct rate than KD-Tree. That is to say, the proposed method has the best comprehensive retrieval performance compared to LSH and KD-Tree. Firstly, in order to verify that the SIFT operator itself has sufficient ability to represent local features of the image, it is used in the method of object-based image retrieval proposed herein. In this paper, the resolution of the two-dimensional plane (image) is 18467×7818 pixels, and the picture size is 413M. We cut it into a small image set of 400 × 400 pixels and then batch extract the global SIFT feature points. At the same time, for each image patch cut out, we put the extracted global feature point set (a total of 284029 128-dimensional SIFT feature points, and the stored file occupies a memory space of 103M) for matching verification.

The experimental results show that the matching accuracy of the features taken from the original image is 100%. This proves that SIFT feature matching achieves good results without external factors (light, distance, viewing angle, size, and user shooting jitter, etc.). Correct feature point detection, extraction, and matching can be performed even for areas where the edges of the image and the image feature points are less taught. It is proved that the SIFT feature points themselves can better solve the problems raised in this paper.

In this index structure experiment, we used the same experimental objects and environments as the above experiments. At the same time, for the upper layer thumbnails, a total of 39,981 128-dimensional SIFT feature points were extracted, and the stored file occupies a memory space of 14.6M.

For the regional image similarity measurement method here, we use the location-based statistical sorting region similarity measure method mentioned in the third chapter. As shown in Table 1, Table 2 and Table 3, we adjust the parameter of the key point contrast threshold in the SIFT feature extraction process to set the value of the feature points to 0.08, 0.085, 0.095, and 0.1 in the original mural image respectively.. 233143, 155153 and 126293 four scale data sets. In these four cases, the total retrieval time and matching accuracy of the 105 images were calculated for the three methods (the number of feature points for the thumbnails mentioned in this article is always 39,981 and the size is 14.6M).

<table>
<thead>
<tr>
<th>Query data set size</th>
<th>Number of feature points in the data set</th>
<th>Match the number of pictures correctly (total of 105 pictures)</th>
<th>Match search time (105 pictures in total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>103M</td>
<td>284029</td>
<td>56</td>
<td>444.4s</td>
</tr>
<tr>
<td>87.3M</td>
<td>233143</td>
<td>60</td>
<td>399.1s</td>
</tr>
<tr>
<td>58M</td>
<td>155153</td>
<td>57</td>
<td>376.9s</td>
</tr>
<tr>
<td>47M</td>
<td>126293</td>
<td>61</td>
<td>368.8s</td>
</tr>
</tbody>
</table>

Table 1. LSH search method image matching experiment results.

<table>
<thead>
<tr>
<th>Query data set size</th>
<th>Number of feature points in the data set</th>
<th>Match the number of pictures correctly (total of 105 pictures)</th>
<th>Match search time (105 pictures in total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>103M</td>
<td>284029</td>
<td>88</td>
<td>130.1s</td>
</tr>
<tr>
<td>87.3M</td>
<td>233143</td>
<td>85</td>
<td>118.7s</td>
</tr>
<tr>
<td>58M</td>
<td>155153</td>
<td>81</td>
<td>117.3s</td>
</tr>
<tr>
<td>47M</td>
<td>126293</td>
<td>82</td>
<td>114.9s</td>
</tr>
</tbody>
</table>

Table 2. KD-Tree retrieval method image matching experiment results.
Table 3. Method image matching experiment results.

<table>
<thead>
<tr>
<th>Query data set size</th>
<th>Number of feature points in the data set</th>
<th>Match the number of pictures correctly (total of 105 pictures)</th>
<th>Match search time (105 pictures in total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>103M</td>
<td>284029</td>
<td>97</td>
<td>114.9s</td>
</tr>
<tr>
<td>87.3M</td>
<td>233143</td>
<td>96</td>
<td>103.5s</td>
</tr>
<tr>
<td>58M</td>
<td>155153</td>
<td>97</td>
<td>102.3s</td>
</tr>
<tr>
<td>47M</td>
<td>126293</td>
<td>97</td>
<td>99.3s</td>
</tr>
</tbody>
</table>

For the problem of partial image retrieval in the large image proposed in this paper, simply using the LSH method will result in excessive memory consumption due to the large size of the image, which will result in slower retrieval speed and even program crash, and return due to LSH. It is a rough search result, and its retrieval accuracy is not ideal.

KD-Tree can solve the problem of K-nearest neighbor better, but only adopt KD-Tree algorithm because the number of feature points is too large, and the dimension of each feature point is too large (128-dimensional), which is not very good in time and precision. Solve the problem raised in this article.

From the table we can clearly see that for the datasets of 4 different situations, the proposed method has obvious advantages compared with the simple KD-Tree and LSH algorithms in both the number of correctly matched pictures and the matching retrieval time. The method proposed in this paper has the best matching retrieval performance.

Acknowledgement

This paper is partly funded by national key R&D program of china research and application of key technology for intelligent dispatching and security early warning of large power grid (No:2017YFB0902600), state grid corporation of china project (No:SGJS00000DKJS1700840) and CERNET Innovation project: immersion VR display system of cultural heritage based on IPV6 (No. NGII20160107).

References


