 Acceleration of similarity-based partial image retrieval using multistage vector quantization

Akisato Kimura, Takahito Kawanishi and Kunio Kashino
NTT Communication Science Laboratories, NTT Corporation
3-1, Morinosato-Wakamiya, Atsugi-shi, Kanagawa, 243-0198, Japan
E-mail : {akisato, kawanisi, kunio}@eye.brl.ntt.co.jp

Abstract

We propose a new method for quick and accurate partial image retrieval from a huge number of images based on a predefined distance measure. The proposed method utilizes vector quantization (VQ) on multiple layers, namely color, block, and feature layers. This can greatly reduce the amount of calculation needed for partial image retrieval. Experiments indicate that the proposed method can detect partial images that are similar to queries through 1000 images within 4 seconds. This is approximately 30 times faster than the method to which multistage VQ is not applied.

1. Introduction

Advances in image capturing devices, storage devices, and infrastructures for broadband networks have made huge numbers of digital images easily accessible and available. Hence, there is an urgent need to develop techniques for retrieving images quickly from large image databases. A common approach is based on keywords. However, providing a keyword for each image is a difficult task for computers and time-consuming for humans.

Hence, content-based image retrieval (CBIR) that finds similar images based on quantitative features such as colors, shapes, and textures, has been investigated [1, 2, 3]. In recent years, region-based approaches [4, 5], which retrieve images that are partially similar to queries, have been widely investigated. A consideration of partial similarities enables CBIR to be more robust and flexible than whole image retrieval. Although some recent work [6, 7, 8] has attempted to decompose an image into individual objects, most approaches use domain-specific constraints to improve the accuracy.

Here, we address the retrieval of partial images similar to a given image (a query) from a large number of images (a database) based on a predefined distance measure $d(\cdot, \cdot)$. Specifically, our task is to find similar partial images such that distance from queries fall below a predefined value (a search threshold) $\theta$, that is,

$$d(D_i(x, y), Q) \leq \theta,$$

where $D_i(x, y)$ is a portion of an image $D_i$ in the database $D = \{D_i\}_{i=1}^M$ at position $(x, y)$ and $Q$ is a query image. We call this framework similarity-based partial image retrieval. A crucial problem related to this task is that finding partial similarities generally requires a huge amount of calculation and storage space for indexes. We have proposed a fundamental algorithm called Sparse-index-based Partial Image REtrieval (SPIRE) [9], which reduces the size of indexes needed for similarity-based partial image retrieval. SPIRE basically extends an indexing algorithm that is almost the same as that of GeneralMatch [10], and applies it to partial image retrieval. Therefore, as is the case of GeneralMatch, SPIRE can theoretically guarantee the same accuracy as exhaustive matching. However, a huge amount of calculation is still needed because of the high-dimensionality of features.

Therefore, in this work, we introduce multistage vector quantization (multistage VQ), which performs vector quantization on multiple layers, namely color, block, and feature layers, to reduce the amount of calculation and the size of the features needed for partial image retrieval. VQ on the color layer (color VQ) is performed for color subtraction. VQ on the block layer (block VQ) is designed to reduce the amount of calculation needed for matching images. VQ on the feature layer (feature VQ) can be viewed as global feature clustering. We can construct an index structure that does not suffer from the “curse of dimensionality” phenomenon.

2. SPIRE

2.1. Outline

Figure 1 outlines the SPIRE algorithm.
Firstly, a block $B$ with a predefined size of $b_x \times b_y$ pixels is extracted from each image $D_i$ of a database $D$, and a block feature $f^{(B)}_D(x_{Di}, y_{Di})$ is extracted from each block. Blocks are extracted while sliding 1 pixel for each block. Secondly, a matching window $W$ with a predefined size of $w_x \times w_y$ pixels is placed on each image at a predefined spacing of $(m_x, m_y)$ pixels, and a matching region $D^{(W)}_i(x_{Di}, y_{Di})$ is extracted from the matching window. We call $(m_x, m_y)$ the margin. These margins must be smaller than the sizes of the matching regions. Thirdly, a matching feature $f^{(W)}_D(x_{Di}, y_{Di})$ is extracted from each matching region, which is a set of block features extracted from disjoint blocks in the matching region. For simplicity, let us suppose that the size of the matching windows is an integer multiple of the size of the blocks. The extracted matching features are stored in some kind of index structure. The procedure described above does not need a query image, and can therefore be performed in advance.

For a given query image $Q$, a matching window, the same size as that of the database, is placed on the query image and matching features $f^{(W)}_Q(x_{Q}, y_{Q})$ are extracted. Unlike the database case, sets of disjoint matching regions are extracted from the query. The query is divided into $\left\lfloor \frac{q_x}{w_x} \right\rfloor \times \left\lfloor \frac{q_y}{w_y} \right\rfloor$ disjoint matching windows. Then, the set of matching windows is slid from position $(0, 0)$ to position $(m_x - 1, m_y - 1)$ in the query. From the above discussion, the query sizes must be larger than the sums of the margins and the sizes of the matching regions. Next, matching regions that have the potential to be included in partial images that are similar to the query are extracted from the database by using the index mentioned above. Lastly, partial images corresponding to the selected matching regions are matched with the query as follows:

$$
d (D_i(x, y), Q) \overset{\text{def}}{=} d \left( f^{(B)}_D(x, y), f^{(W)}_Q(x_{Q}, y_{Q}) \right) \overset{\text{def}}{=} \sum_{i=0}^{\frac{q_x}{w_x} - 1} \sum_{j=0}^{\frac{q_y}{w_y} - 1} d \left( f^{(B)}_D(x + ib_x, y + jb_y), f^{(W)}_Q(ib_x, jb_y) \right). \tag{1}$$

2.2. Pruning matching regions

This section describes how relevant matching regions are selected by utilizing the index created from the database.

All relevant matching features of the database are selected such that the distance between the matching feature of the query falls below a predefined value (a pruning threshold) $\hat{\theta}$. The distance between features is calculated in the same way as Eq. (1), as follows:

$$
d \left( f^{(W)}_D(x_{Di}, y_{Di}), f^{(W)}_Q(x_{Q}, y_{Q}) \right) \overset{\text{def}}{=} \sum_{i=0}^{\frac{q_x}{w_x} - 1} \sum_{j=0}^{\frac{q_y}{w_y} - 1} d \left( f^{(B)}_D(x_{Di} + ib_x, y_{Di} + jb_y), f^{(W)}_Q(ib_x, jb_y) \right),$$

From Eqs. (1) and (2), we can obtain

$$
d \left( f^{(W)}_D(x_{Di}, y_{Di}), f^{(W)}_Q(x_{Q}, y_{Q}) \right) \leq d \left( f^{(W)}_D(x_{Di} + x_{Q}, y_{Di} + y_{Q}), f^{(W)}_Q(x_{Q}, y_{Q}) \right).$$

It is important to determine an appropriate pruning threshold to avoid false dismissals. We set the pruning threshold as $\hat{\theta} = \frac{\theta}{N}$, where $N$ is the minimum number of disjoint matching regions of database images in windows with the same size as the query. For simplicity, we suppose that the size of the matching windows is an integer multiple of the margin. Then, $N$ is calculated as follows:

$$
N_x = N_y, \\
N_i = \max \left( 1, \left\lfloor \frac{q_i}{w_i} \right\rfloor - 1 \right), \quad (i = x, y)
$$

The following lemma is obtained from the relationship between the search threshold and the pruning threshold:

**Lemma 1**

If $d(D(x_{Di}, y_{Di}), Q) \leq \theta$, there exists $(x_{Q}, y_{Q})$ such that

$$
d \left( f^{(W)}_D(x_{Di} + x_{Q}, y_{Di} + y_{Q}), f^{(W)}_Q(x_{Q}, y_{Q}) \right) \leq \hat{\theta}.$$
This means that there is no false dismissal if the pruning threshold is appropriately determined. We can easily obtain Lemma 1 from Theorem 1 in [10].

3. Proposed method

3.1. Color VQ and block VQ

Color VQ and block VQ accelerate feature matching by refining feature representation. In the following, we describe how to compose features.

Figure 2 outlines the procedure. First, color VQ is performed. Each pixel is quantized based on a color palette created beforehand, and then subtractive color images are created. Next, block VQ is performed on each feature. Features are quantized based on a VQ codebook created beforehand by a certain learning algorithm, such as the LBG algorithm. The distances between typical block patterns are calculated in advance, and stored with the VQ codebook. A feature for matching (called a matching feature) is composed of a set of indexes, each of which corresponds to a typical block pattern. We can quickly calculate the distance between matching features by using the distances between typical block patterns as follows:

\[
d(f_D(x_D, y_D), f_Q) = \frac{1}{2} \left( \sum_{x=1}^{w_x} \sum_{y=1}^{w_y} d(idx_D(x_D + x, y_D + y), idx_Q(x, y)) \right),
\]

where \(idx_D\) and \(idx_Q\) are indexes of block VQ.

It is not necessarily the case that utilizing these features gives the same search results as the original features. However, this provides robustness with respect to certain kinds of noise or distortion.

3.2. Feature VQ

Feature VQ is utilized in the indexing and pruning stages to reduce the number of matching calculations. In creating indexes, matching features are quantized and classified based on a VQ codebook created in advance.

When selecting relevant features, we apply the Global Pruning algorithm [11] to the proposed method. We briefly describe Global Pruning below.

It should be noted that this pruning technique generates no approximation when a proper condition is satisfied. Figure 3 illustrates the situation where a \(w_x \times w_y\)-dimensional feature space is sliced by a plane on which three specific points, \(Q\), \(C_1\), and \(C_2\), reside simultaneously, where \(Q\) is the query feature, \(C_1\) the centroid of the cluster (the initial cluster) whose centroid is the minimum distance from the feature, and \(C_2\) is the centroid of another cluster (the target cluster). When the minimum distance between the query feature and the target cluster, \(d_\theta\), exceeds the pruning threshold, features in the target cluster must be matched. Then, we determine \(d_\theta\). We consider the following distance relationship:

\[
h^2 = d_{Q1}^2 - \left( \frac{1}{2} d_{12} - d_\theta \right)^2 = d_{Q2}^2 - \left( \frac{1}{2} d_{12} + d_\theta \right)^2,
\]

where \(d_{Q1}\) is the distance between the query feature and the centroid of the initial cluster, \(d_{Q2}\) is the distance between the query feature and the centroid of the
target cluster, and $d_{12}$ is the distance between the centroids of those clusters. Then, we obtain

$$d_\theta = \frac{d_{Q2}^2 - d_{Q1}^2}{2d_{12}}.$$  

4. Experiments

4.1. Conditions

We tested the proposed method in terms of search speed. The tests were carried out on a PC (Intel Xeon 2.8GHz). In the experiments, we used a real-life image data set [13, 14] from Pennsylvania State University. This data set contains 1000 images stored in JPEG format, with a size of 384 × 256 pixels. We chose 10 portions with a size of $q_x \times q_y = 80 \times 80$ pixels from the images as queries. The matching regions were those with a size of $w_x \times w_y = 64 \times 64$ pixels and located at a spacing of $m_x \times m_y = 16 \times 16$ pixels in each database image. Each pixel was quantized to 64 representative colors, and each block was quantized to 64 representative blocks. The block size was 8 × 8 pixels. Thus, the dimensionality of the features was 64 (= 8 × 8).

4.2. Results

We examined the relationship between the number of clusters of feature VQ and the search time. The baseline search method was the Skipping Template Matching method [12], which was proposed earlier. Figure 4 shows the result. The horizontal axis indicates the search methods. The figure shows that the proposed method reduces the search time by introducing multi-stage VQ.

5. Conclusion

We have proposed a new method for quick partial image retrieval, which is a task that involves finding images partially similar to a query from a huge number of database images. The proposed method utilizes multi-stage vector quantization, that is, color VQ, block VQ, and feature VQ. This greatly reduces the amount of calculation required when searching. Experiments showed a significant reduction in the search speed.

References