3
Bag-of-visual-Word (BoW) Representation

So far, we discussed how to extract features that are robust under various transformation. For each image, we can extract a different number of features. This variable number of image features pose a problem for matching between two images for image search. To address this issue, we discuss Bag-of-visual-Word (BoW) model. We also discuss inverted index to efficiently identify similar images out of an image database containing many images.

3.1 Bag-of-visual-Word (BoW) Model

Suppose that we extract a set of local features, e.g., SIFTs or deep features from multiple regions, for images. When we have such a model representation whose size varies, it is rather difficult to measure the similarity between those varying representations. Furthermore, it complicates many other parts, e.g., indexing structures, of image search approaches. As a result, it is common to have a representation that has a fixed dimensionality that represents those local features.

As a representation with the fixed dimensionality, the Bag-of-visual-Word (BoW) model is introduced. Since computing such representations from multiple local features is known as a pooling or aggregation operations, BoW is considered as a pooled or aggregated representation.

The bag-of-visual-words model is simply a histogram of visual words computed from local image features, e.g., SIFTs, where visual words are commonly appearing features, i.e., clustered features.

The BoW model of an image is a histogram of visual words from those low-level features. For computing the BoW feature from an image, we first compute visual words, i.e., code words. A common way of computing such visual words is to compute clusters from an image database using a clustering method such as k-means clustering. We can conceptually treat each cluster to represent a visual word, and thus those computed clusters serves as visual vocabulary. Fig. 3.1 shows an example of computed visual words extracted from hundreds of images; this image is excerpted from [1].

Once we compute such visual words as clusters, we then compute

[1] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005

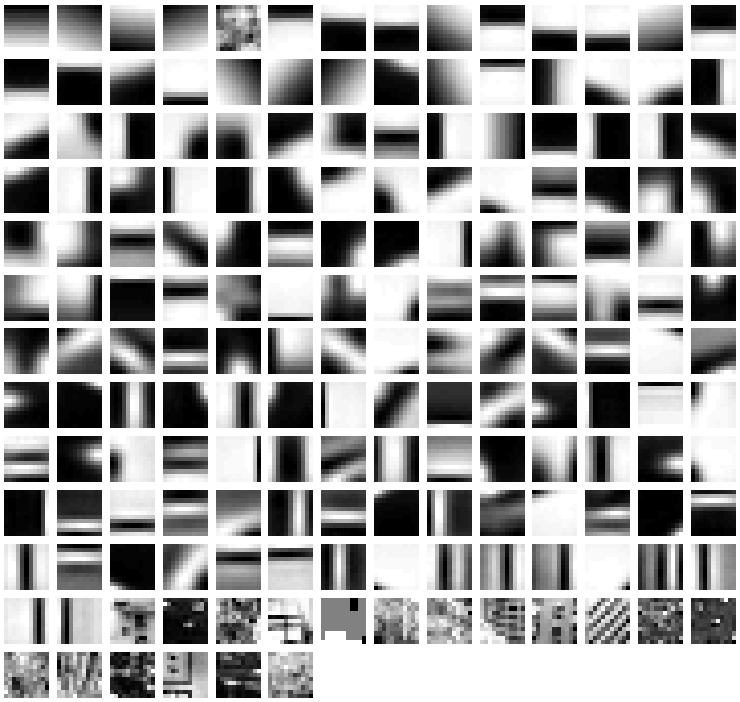

Figure 3.1: This figure shows an example of a codebook computed from hundreds of images that are from 13 categories. These image patches are extracted in a sliding window manner with random scaling. Clusters are sorted in an decreasing order in terms of occurrence from the top-left to the bottom. This image is excerpted from the Fei-Fei' 2005 paper.

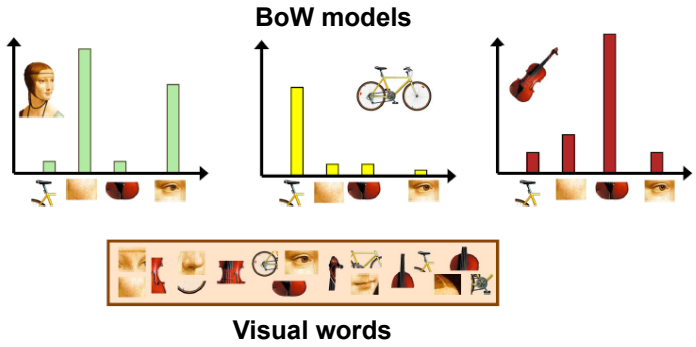

Figure 3.2: This figure shows conceptual illustration of computed visual words on the bottom, which are commonly appearing image patches in this example. Also, each image is represented by a Bag-of-visual-Word (BoW) model, which is a histogram encoding the number of occurrence of each visual word. This image is excerpted from the slide of Fei-Fei.

the histogram of visual words by checking the assignment of each feature of an image to clusters. Fig. 3.2 shows conceptual illustration of computed visual words on images and the computed histograms for each example image.

For choosing a cluster from a feature, we can simply identify a nearest cluster from the feature by measuring the (L2) distance between the feature and the centroids of clusters. Based on this process, we can compute the BoW feature from a set of local features from a image, where the dimensionality of the BoW feature linearly increases the number of visual words, i.e., clusters.

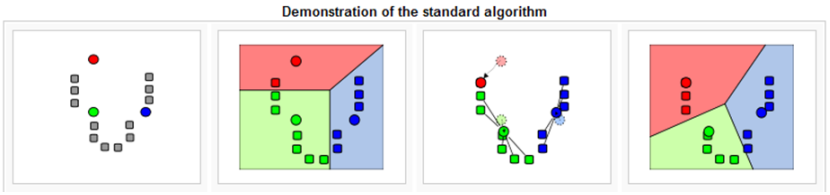

Figure 3.3: This figure illustrates the process of k-means clustering. From the left, we choose three random seeds and then compute initial clusters. We then update the mean of each cluster and perform the assignment process again. This figure is excerpted from wiki.

K-means clustering. k-means clustering is a simple, unsupervised method of computing k clusters; this is also known as Lloyd's algorithm. Formally speaking, given n data, denoted by x_j, we aim to find k clusters, i.e., $\mathbb{C} = \{C_1, \cdots, C_k\}$, in a way to minimize the variance within each cluster. In other words, these k clusters are computed to minimize the following objective function:

$$\underset{\mathbb{C}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2, \tag{3.1}$$

where μ_i is the mean of the i-th cluster.

Unfortunately, this simple problem turns out to be NP-hard [2].

Nonetheless, there is a simple heuristic approach that can compute a reasonable solution. This method, k-means clustering, starts with an input of the number of clusters k. Initially, we need to have an initial set of clusters. One can simply use randomization, i.e., randomly pick k data and treat them as k different clusters. We then perform the following steps iteratively:

1. **Assignment:** For each data x_j, we assign it to the closest cluster among k clusters based on the L2 distance between x_i and μ_i the mean of each cluster.

2. **Update:** We update means of clusters based on the new assignment of data to clusters.

An example of k-means clustering is shown in Fig. 3.3.

k-means clustering is fast for a small number of data, but is not scalable for massive amount of data. Furthermore, k-means clustering assumes isotropic cluster shapes for clusters. For more general shapes, the EM technique can be used (Ch. 16.1).

Overall search process. Given an image database, we pre-compute BoW models for all the images in the database. When a query image I_q is given, we also compute the BoW model for the query image and then go over each image, I_i, in the database and measure the L2 distance between the BoW models of the query image I_q and the DB image I_i.

[2] When a problem C is NP-hard, every of known NP problems whose candidate answers can be checked in a polynomial time can be reduced to the problem C. It is believed that NP-hard problems cannot be solved in a polynomial time.

Once we go over all the images in the DB, we choose a set of small images, i.e., shortlist, that have smallest L2 distance out of all the DB images, and show the shortlist to the user as the final result. This simple process can identify a set of small images that have similar BoW features to that of the query image, but can be extremely slow, especially when the DB has many images, say billions of images. We discuss an inverted index structure to accelerate the overall process in Ch. 3.4. Also, using the L2 distance may not be the best choice. We discuss other alternatives on distance measures, which serve as similarity measure between two image features (Ch. 3.2).

TF-IDF. TF (Term frequency) and IDF (Inverse document frequency) have been widely used for text search, and these concepts are adopted for image search techniques. TF-IDF is intuitive concept and is extended into other various normalization techniques.

Suppose that you want to identify images containing apples, and two images are searched as one image I_A contains only one image patch containing the apple, while the other image I_B is identified to have two different image patches containing the apple. Which one do you prefer to view first? Most users are likely to see I_B containing two apples first before browsing I_A, assuming that I_B can have image information that are suitable to the given query text, apple. To realize this concept, we use TF that give a higher weight as we have more occurrence of a term, i.e., visual word, in the feature. Actually, our BoW model already adopts the TF concept, since we count the number of occurence of each visual word by using the histogram.

Now suppose that you want to search images containing apples and oranges. Assume that all the images in the DB have the visual word of oranges. Is then the query keyword of orange useful? Maybe not, since the keyword of orange is universal ad thus it does not have any discriminative power in this example.

To support this intuition as well as TF, we use the following TF-IDF weight for i-th visual word"

$$w_i^{IDF} = \log \frac{N}{n_i}, \tag{3.2}$$

where N and n_i are the number of images in the DB and the number of images containing i-th visual word, respectively. We can multiply this weight for each visual word to the BoW model. When n_i becomes N, w_i^{IDF} becomes zero, thus the visual word does not have any representation information.

Recall and precision. It is important to evaluate different search techniques and see which is better than the others. For evaluating

the quality, i.e., accuracy, of a search algorithm, we commonly use two measures: recall and precision. Suppose that G and R represent ground-truth set of images and result set given a query, respectively.

Recall measures how much of images of the ground-truth set is identified from the result set, and is defined as the following:

$$Recall = \frac{|G \cap R|}{|G|}. \tag{3.3}$$

On the other hand, precision measures how much of images of the result set is from the ground truth set:

$$Precision = \frac{|G \cap R|}{|R|}. \tag{3.4}$$

Usually, as we have a longer shortlist, we can have more a higher recall value, but we tend to have a lower precision. As a result, depending on the length of the shortlist, the recall and precision values can vary. To address this issue, we can measure different precision values by varying the shortlist size and measure their average, which is known as mean average precision (mAP).

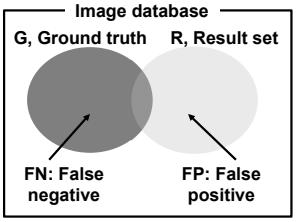

Figure 3.4: This shows ground-truth, G, and result set, R, given a query.

Limitations of BoW models. The BoW model identifies important image patches as visual words and describes the given image with those visual words. Nonetheless, the BoW model does not describe their spatial relationship between those visual words. As a result, the model has drawbacks on representing other information like spatial relationship between important features of images. Ch. 3.3.1 discuss how to represent those spatial relationship and utilize for accurate image search.

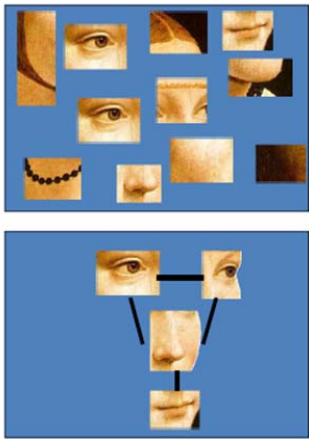

Figure 3.5: The BoW model identifies important patches on the image, but does not describe their spatial relationship as shown in the bottom. Image credit to Fei-Fei Li.

3.1.1 *Commonly Asked Questions*

We need to define the number of words for the bag-of-visual-words approach. Is there any automatic way of defining it? As far as I know, there have not been many approaches on defining it in an automatic way. Many papers attempted different numbers of visual words, and measured the accuracy and the memory requirement as a function of the number of visual words. Then it chooses a particular number of words that gives a reasonable accuracy with a memory requirement for their problems.

3.2 *Similarity Measure*

Once the BoW feature or some other feature is computed for each image, we can use various similarity measures to see how an image is

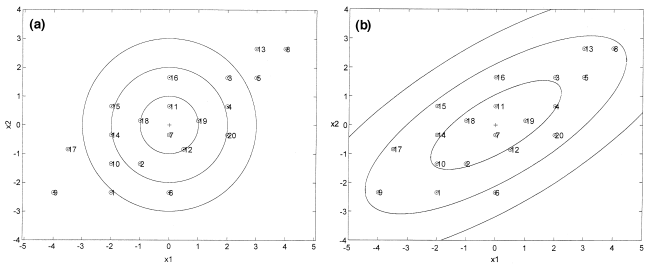

Figure 3.6: This illustrates iso-distances from the center point given the L_2 distance and Mahalanobis distance on the left and right.

similar to others. A well-known similarity measure of a difference X is L_p-norm:

$$\|X\|_p = \left(\sum_i X_i^p \right)^{1/p}, \tag{3.5}$$

where i is an index for each dimension of the difference vector X. L_1 and L_2-norms are commonly used; for the sake of simplicity, $||X||$ indicates L_2-norm, unless mentioned otherwise. L_p-norm maps to other many well-known functions, where L_∞ is the maximum function. Another well known function is a dot product between two vectors that measures the angle between them:

$$a \cdot b = \|a\| \, \|b\| \cos(\theta), \tag{3.6}$$

where θ is the angle between two vectors a and b.

These L_p measures treat each dimension of feature vectors equally. In some cases, those dimensions can be correlated each other. Furthermore, some data can be distributed more widely in a dimension compared to other dimensions. To consider such anisotropic distributions, the Mahalanobis distance metric between two vectors $\mathbf{X}$ and $\mathbf{Y}$ is used:

$$D_m(\mathbf{X}, \mathbf{Y}) = \sqrt{\mathbf{D}^T \mathbf{S}^{-1} \mathbf{D}}, \tag{3.7}$$

where $\mathbf{D} = \mathbf{Y} - \mathbf{X}$ and $\mathbf{S}$ is the covariance matrix, where $\mathbf{S}_{ij} = E[(\mathbf{X}_i - \mu_i)(\mathbf{Y}_j - \mu_j)]$.

Fig. 3.6 intuitively illustrates the equal distances from the center point given the Euclidean distance and Mahalanobis distance; the image is excerpted from [3]. Note that Mahalanobis distance considers the variance of the data and principal components of the data, while the Euclidean distance treats data in the same distance equally. As a result, the Mahalanobis distance can be better than the Euclidean distance in data with anisotropic distribution.

More detailed discussions on learning the similarity measures is available in Ch. 10.

[3] R. De Maesschalck, D. Jouan-Rimbaud, and D.L. Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1 – 18, 2000

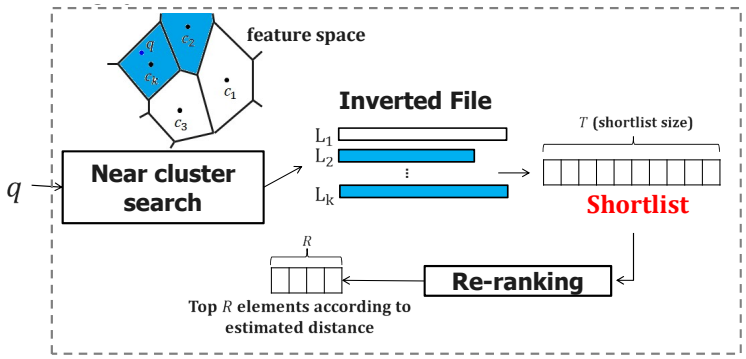

Figure 3.7: This figure shows a schematic overview of an image search framework that we discuss in this chapter. Given a query image, we first compute a shortlist containing an initial ordered list of similar images, which is then re-ranked for computing the final results. This image is courtesy of JaePil Heo.

3.3 *Re-ranking*

Given a query image, we first identify a set of initial results that are similar to the query. Typically, the set of initial results are significantly smaller than the image list in the database. As a result, this set is called shortlist. Fig. 3.7 shows an schematic overview of image search, where we compute a shortlist and re-rank the list for computing the final results; inverted file is discussed in Ch. 3.4.

Once we compute the shortlist, we can return this result to users, but improve it by performing additional operations as postprocessing. In this section, we discuss two well known postprocessing techniques, query expansion and spatial verification.

Note that these postprocessing techniques rerank images only in the shortlist. As a result, whatever we do for postprocessing, we cannot improve the recall of the result. Therefore, for computing the shortlist, we typically aim to achieve a high recall, while the postprocessing step improves the precision.

3.3.1 *Spatial verification*

Once we compute a shortlist from the query, we can employ additional operations to improve the matching quality. One of most commonly approaches is to perform spatial verification that checks how well matching features in the query and an image in the shortlist are spatially aligned. Fig. 3.8 shows matching results between two images without performing the spatial verification; the image is excerpted from [4]. When we do not consider spatial relationship between features, we may get very incorrect matching between features of images.

RANSAC (RANdom SAmple Consensus) is one of common techniques for performing the spatial verification. Its main idea is to estimate a hypothesis on a possible transformation between two images and then to test how well the hypothesis works well with other

[4] G. Shi, X. Xu, and Y. Dai. Sift feature point matching based on improved ransac algorithm. In *International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2013

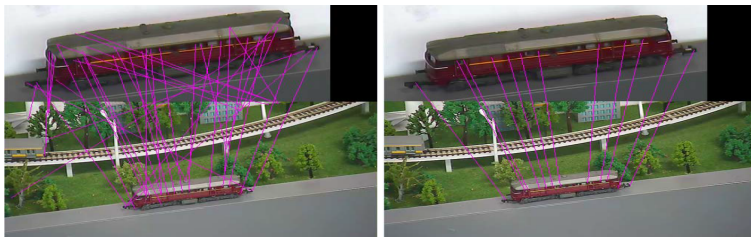

Figure 3.8: The left figure shows initial matching points between two images, while the right shows matching of inliers computed by RANSAC. These images are excerpted from Shi's paper.

data. For RANSAC, we assume a particular linear transformation between two images. One can use a projective transformation, which can be represented by 3 by 3 matrix, where we have eight degrees of freedom.

RANSAC is a common method for performing the spatial verification that iteratively sets up a hypothetical transformation and validates it.

Specifically, given a list of matching points between two images, RANSAC iteratively performs the following operations:

1. **Setting up a hypothesis.** Given the projective transformation, we randomly choose a subset of matching points to estimate the transformation. This serves as a hypothesis for the transformation between two images.

2. **Validation.** We then validate the hypothesis based on the rest of the matching points. If our hypothesis is a reasonable one, there would be many inliers that follows the hypothetical transformation.

We perform these steps until we found a reasonable set of inliers. Otherwise, we treat two images to be dissimilar in terms of the spatial verification. The right image of Fig. 3.8 shows inlier matching points between two images. Since RANSAC is based on random sampling on matching points, it can be quite slow. Nonetheless, it has been widely used for performing the spatial verification.

3.3.2 *Query expansions*

Query expansion is adopted from text retrieval. Its main idea is that once an initial set of images is acquired from the query, we then use the shortlist as additional queries, for achieving a higher recall. This approach, however, works well when the initial results contain correctly matched images to the query.

To utilize correctly matched images, we use only spatially verified images and use them as additional queries. Also, there are many different ways of using those verified images. One simple, yet effective method is to use an average image that is computed by averaging features (e.g., BoW models) of the verified images [5].

[5] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, pages 1 –8, oct. 2007

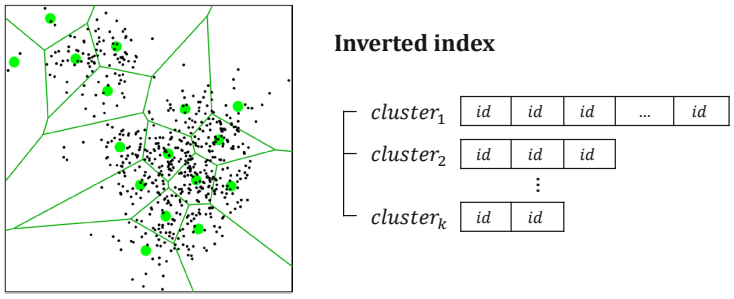

Figure 3.9: This figure shows the inverted index that is computed from the clusters defined in a feature space. The image is the courtesy of Zhe Lin.

3.4 *Indexing Schemes*

So far, we simply assumed that once we compute a feature from the query image, we then go through every single image in the image database and compute the top-k similar images. This exhaustive approach may work in a small-scale image database, but should be prohibitively slow for a large-scale image database consisting of billions of images. As a result, it is critical to access only a subset of images in the database for identifying similar images given the query image.

For efficiently identifying similar images given the query image, the inverted index, also known as inverted file, is commonly used. Its main idea is to partition images of the database into a set of clusters and to pre-compute an inverted index for each cluster, c_i that lists images that belong to the cluster c_i. Fig. 3.9 shows a schematic illustration of the inverted index.

At a query phase, we first identify a set of clusters that are close to the query. For this process, we can perform nearest neighbor search given the query, which is discussed in Ch. 4. For each cluster of the identified clusters, we access its corresponding inverted index, where we can access images that belong to that cluster. These images are candidates for similar images to the query image. As a result, it is likely that as we identify more clusters, we can improve the recall of our search, while spending more time to search similar images.

As an image information in the inverted index, we typically record its image descriptor and its ID. The image descriptor is necessary for computing the similarity measure between the image and the query image. Also, when we decide to show its content of the image to users, we need to access its image file, and thus need its image ID.

When we consider a large scale image database, the memory requirement of the inverted index can be very high. For example, suppose that we use 4k dimensional space for the image descriptor. When we have one billion images in the database, the memory space only for the image descriptor in the inverted index can be

higher than 4 TB, which cannot be stored in the main memory. As a result, to reduce down its memory requirement, we use a compact representation, which is discussed in Ch. 4.