search by Chum and colleagues [CPZ08]. Jain *et al*. devise an algorithm to generate semi-supervised hash functions that support learned Mahalanobis metrics or kernels [JKG08], while Kulis and Grauman provide LSH functions amenable to arbitrary kernel functions [KG09].

Embedding functions offer another useful way to map expensive distance functions into something more manageable computationally. Recent work has considered how to construct or learn an embedding that will preserve the desired distance function, typically with the intention of mapping to a very low-dimensional space that is more easily searchable with known techniques [AASK04, TFW08, SH07, WTF09]. These methods are related to LSH in the sense that both seek small "keys" that can be used to encode similar inputs, and often these keys exist in Hamming space. While most work with vector inputs, the technique in [AASK04] accepts generic distance functions.

In short, all such search and embedding methods offer ways to reduce the computational cost of finding similar image descriptors within a large database. The appropriate choice for an application will depend on the similarity metric that is required for the search, the dimensionality of the data, and the offline resources for data structure setup or other overhead costs.

5.2 Visual Vocabularies and Bags of Words

In this section we overview the concept of a *visual vocabulary*—a strategy that draws inspiration from the text retrieval community and enables efficient indexing for local image features. We first describe the formation of visual words (Section 5.2.1), and then describe their utility for indexing (Section 5.2.2) and image representation (Section 5.2.3).

5.2.1 Creating a Visual Vocabulary

Methods for indexing and efficient retrieval with text documents are mature, and effective enough to operate with millions or billions of documents at once. Documents of text contain some distribution of words, and thus can be compactly summarized by their word counts (known as a bag-of-words). Since the occurrence of a given word tends to be sparse across different documents, an index that maps words to the files in which they occur can take a keyword query and immediately produce relevant content.

What cues, then, can one take from text processing to aid visual search? An image is a sort of document, and (using the representations introduced in Chapter 3)

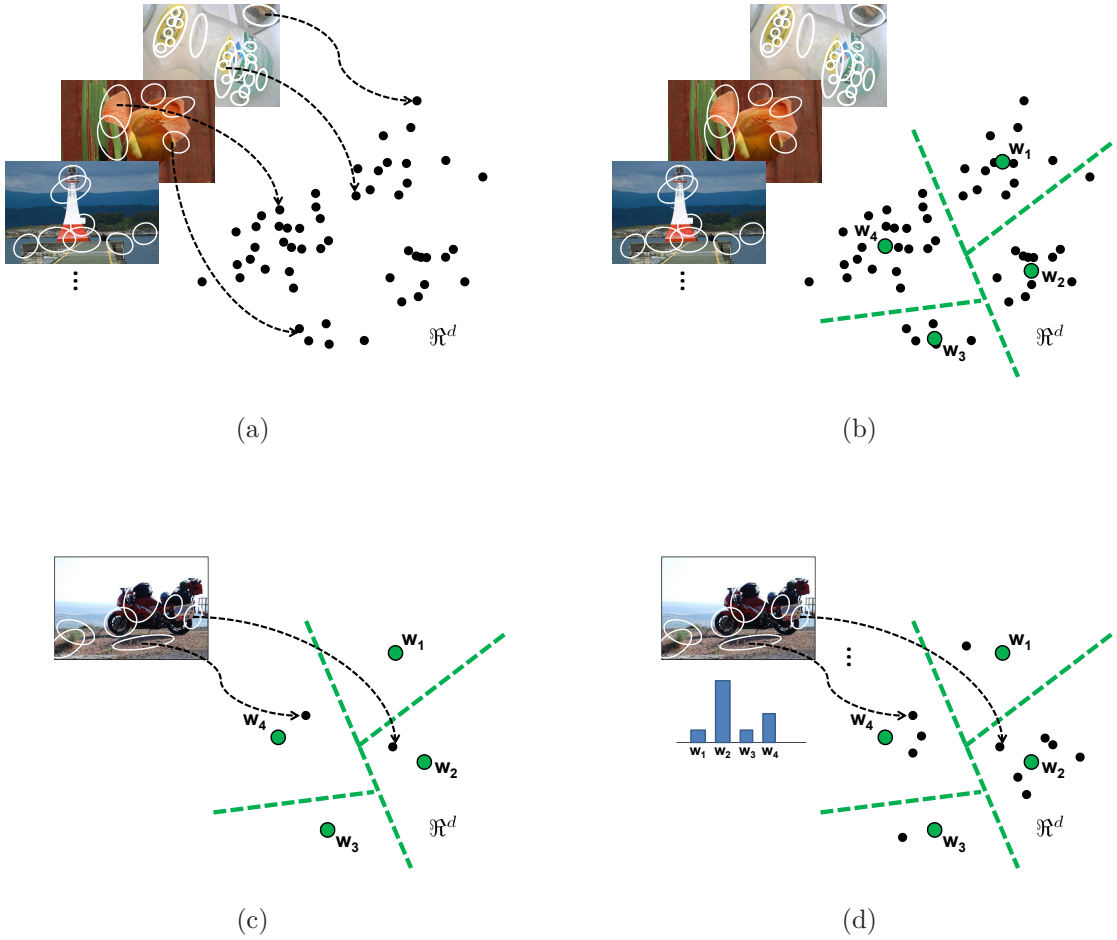


Figure 5.2: A schematic to illustrate visual vocabulary construction and word assignment. (a) A large corpus of representative images are used to populate the feature space with descriptor instances. The white ellipses denote local feature regions, and the black dots denote points in some feature space, *e.g.*, SIFT. (b) Next the sampled features are clustered in order to quantize the space into a discrete number of visual words. The visual words are the cluster centers, denoted with the large green circles. The dotted green lines signify the implied Voronoi cells based on the selected word centers. (c) Now, given a new image, the nearest visual word is identified for each of its features. This maps the image from a set of high-dimensional descriptors to a list of word numbers. (d) A bag-of-visual-words histogram can be used to summarize the entire image. It counts how many times each of the visual words occurs in the image.

it contains a set of local feature descriptors. However, at first glance, the analogy would stop there: text words are discrete "tokens", whereas local image descriptors are high-dimensional, real-valued feature points. How could one obtain discrete "visual words"?

To do so, we must impose a quantization on the feature space of local image descriptors. That way, any novel descriptor vector can be coded in terms of the (discretized) region of feature space to which it belongs. The standard pipeline to form a so-called "visual vocabulary" consists of (1) collecting a large sample of features from a representative corpus of images, and (2) quantizing the feature space according to their statistics. Often simple k-means clustering is used for the quantization; the size of the vocabulary k is a user-supplied parameter. In that case, the visual "words" are the k cluster centers. Once the vocabulary is established, the corpus of sampled features can be discarded. Then a novel image's features can be translated into words by determining which visual word they are nearest to in the feature space (*i.e.*, based on the Euclidean distance between the cluster centers and the input descriptor). See Figure 5.2 for a diagram of the procedure.

Drawing inspiration from text retrieval methods, Sivic and Zisserman first proposed quantizing local image descriptors for the sake of rapidly indexing video frames with an inverted file [SZ03]. They showed that local descriptors extracted at interest points could be mapped to visual words by computing prototypical descriptors with k-means clustering, and that having these tokens enabled faster retrieval of frames containing the same words. Csurka and colleagues first proposed using quantized local descriptors for the purpose of object categorization; an image's descriptors are mapped to a bag-of-words histogram counting the frequency of each word, and categories are learned using this vector representation [CBDF04].

Prior to that work, researchers had considered a sort of visual vocabulary specifically for the texture classification problem. Leung and Malik proposed quantizing the densely sampled outputs of filter banks to form a vocabulary of *textons*, which then allowed an image of a material to be summarized with a histogram of texton occurrences [LM99]. Later extensions showed how to improve invariance properties and flexibility to viewing conditions of the materials [CD01, VZ02].

What will a visual word capture? The answer depends on several factors, including what corpus of features are used to build the vocabulary, the number of words selected, the quantization algorithm used, and the interest point or sampling mechanism chosen for feature extraction. In general, patches assigned to the same visual word should have similar low-level appearance (see Figure 5.3). Particularly when the vocabulary is formed in an unsupervised manner, there are no constraints that the common types of local patterns be correlated with object-level parts. However, in Chapter 7 we will see some methods that use visual vocabularies

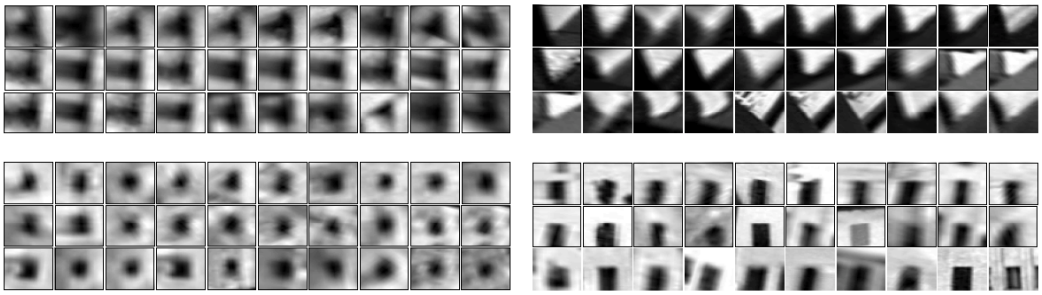


Figure 5.3: Four examples of visual words. Each group shows instances of patches that are assigned to the same visual word.*KG: images from Sivic 2003*

or codebooks to provide candidate parts to a part-based category model.

The discussion above assumes a flat quantization of the feature space, but many current techniques exploit hierarchical partitions [NS06, GD05b, GD06, MTJ06, YLD07, BZM07]. Nister and Stewenius proposed the idea of a *vocabulary tree*, where one chooses a branching factor and number of levels, and then uses hierarchical k-means to recursively subdivide the feature space. Vocabulary trees offer a significant advantage in terms of the computational cost of assigning novel image features to words—from linear to logarithmic in the size of the vocabulary. This in turn makes it practical to use much larger vocabularies (*e.g.*, on the order of one million words). Experimental results suggest that these more specific words (smaller quantized bins) are particularly useful for matching specific instances of objects [NS06, PCI$^+$07, PCI$^+$08]. Since quantization entails a hard-partitioning of the feature space, it can also be useful in practice to use multiple randomized hierarchical partitions, and/or to perform a soft assignment in which a feature results in multiple weighted entries in nearby bins. Recent work considers how to hierarchically aggregate the lowest level tokens from the visual vocabulary into higher level parts, objects, and eventually scenes [STFW05, AT06, PZC09].

An important concern in creating the visual vocabulary is the choice of data used to construct it. Generally researchers report that the most accurate results are obtained when using the same data source to create the vocabulary as is going to be used for the classification or retrieval task. This can be especially noticeable when the application is for specific-level recognition rather than generic categorization. For example, to index the frames from a particular movie, the vocabulary made from a sample of those frames would be most accurate; using a second movie to form the vocabulary should still produce meaningful results, though likely weaker accuracy. When training a recognition system for a particular set of categories, one would typically sample descriptors from training examples covering all categories to try and ensure good coverage. That said, with a large enough pool of features

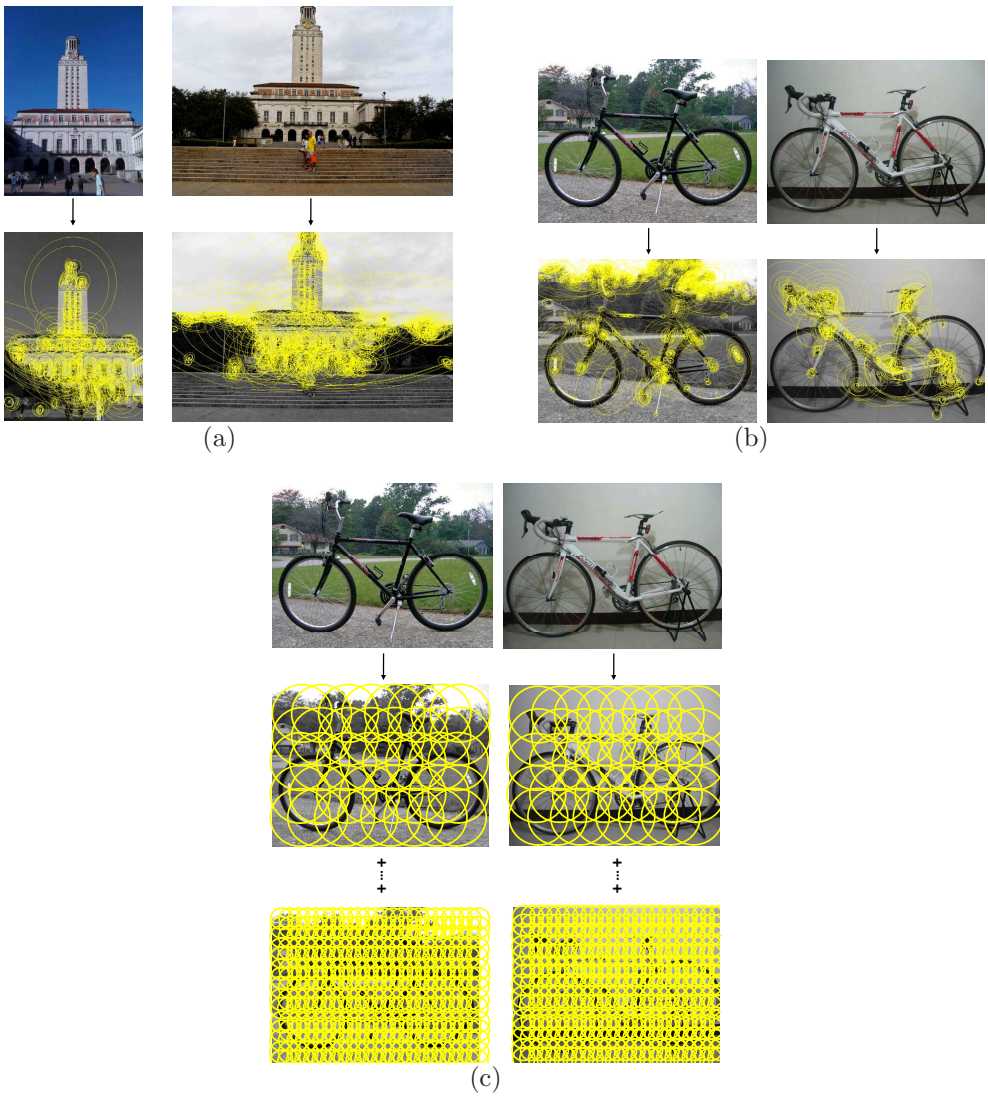


Figure 5.4: Depending on the level of recognition and data, *sparse* features detected with a scale invariant interest operator or *dense* multi-scale features extracted everywhere in the image may be more effective. To match specific instances (like the two images of the UT Tower in (a)), the sparse distinctive points are likely preferable. However, to adequately represent a generic category (like the images of bicycles in (b) and (c)), more coverage may be needed. Note how the interest operator yields a nice set of repeatable detections in (a), whereas the variability between the bicycle images leads to a less consistent set of detections in (b). A dense multi-scale extraction as in (c) will cost more time and memory, but guarantees more "hits" on the object regions the images have in common. The Harris-Hessian-Laplace detector [MS04b] was used to generate the interest points shown in these images.

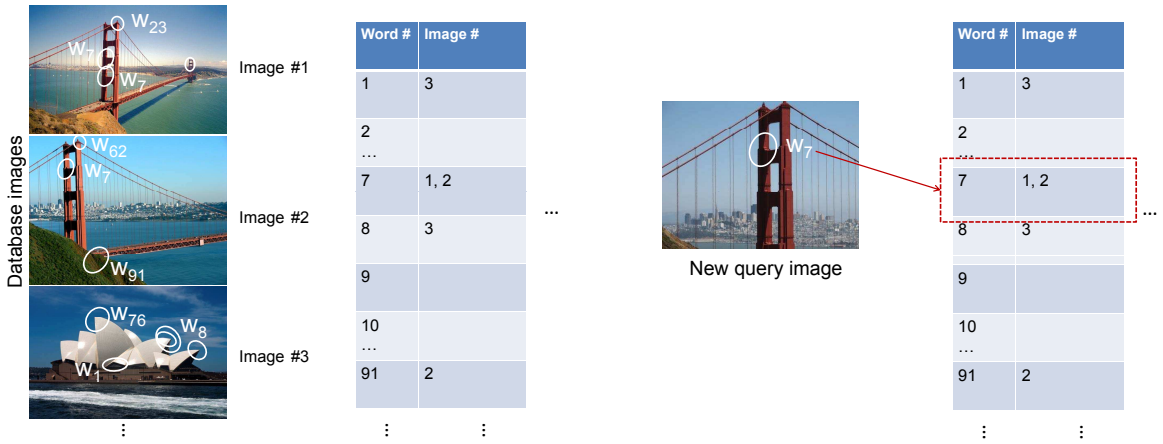
taken from diverse images (admittedly, a vague criterion), it does appear workable to treat the vocabulary as "universal" for any future word assignments. Furthermore, researchers have developed methods to inject supervision into the vocabulary [WCM05, PDCB06, MTJ06], and even to integrate the classifier construction and vocabulary formation processes [YJSJ08]. In this way, one can essentially learn an application-specific vocabulary.

The choice of feature detector or interest operator will also have notable impact on the types of words generated, and the similarity measured between the resulting word distributions in two images. Factors to consider are (1) the invariance properties required, (2) the type of images to be described, and (3) the computational cost allowable. Figure 5.4 visualizes this design choice. Using an interest operator (*e.g.*, a DoG detector) yields a sparse set of points that is both compact and repeatable due to the detector's automatic scale selection. For specific-level recognition (*e.g.*, identifying a particular object or landmark building), these points can also provide an adequately distinct description. A common rule of thumb is to use multiple complementary detectors; that is, to combine the outputs from a corner-favoring interest operator with those from a blob-favoring interest operator.

On the other hand, for category-level tasks, research suggests that a regular, *dense* sampling of descriptors can provide a better representation [NJT06, LSP06], essentially because it means the object has more regular coverage: there is nothing that makes the "interest" points according to an invariant detector correspond to the semantically interesting parts of an object. When using a dense sampling, it is common to extract patches at a regular grid in the image, and at multiple scales. A compromise on complexity and descriptiveness is to sample randomly from all possible dense multi-scale features in the image. See Figure 5.4. Overall, dense features are now more commonly used for category recognition, whereas interest operators are more commonly used to match instances of objects or locations.

5.2.2 Inverted File Indexing

Visual vocabularies offer a simple but effective way to index images efficiently with an *inverted file*. An inverted file index is just like an index in a book, where the keywords are mapped to the page numbers where those words are used. In the visual word case, we have a table that points from the word number to the indices of the database images in which that word occurs. For example, in the cartoon illustration in Figure 5.5, the database is processed and the table is populated with image indices in part (a); in part (b), the words from the new image are used to index into that table, thereby directly retrieving the database images that share its distinctive words.



(a) All database images are loaded into the index mapping words to image numbers.

(b) A new query image is mapped to indices of database images that share a word.

Figure 5.5: Main idea of an inverted file index for images represented by visual words.

Retrieval via the inverted file is faster than searching every image, assuming that not all images contain every word. In practice, an image's distribution of words is indeed sparse. Since the index maintains no information about the relative spatial layout of the words per image, typically a spatial verification step is performed on the images retrieved for a given query (*e.g.*, see [PCI+07]).

5.2.3 Image Representation with a Bag of Visual Words

As briefly mentioned above, the visual vocabulary also enables a compact summarization of all an image's words. The common text description of a "bag-of-words" can be mapped over to the visual domain: the image's empirical distribution of words is captured with a histogram counting how many times each word in the visual vocabulary occurs within it (see Figure 5.2 (d)).

What is convenient about this representation is that it translates a (usually very large) set of high-dimensional local descriptors into a single sparse vector of fixed dimensionality across all images. This in turn allows one to use many machine learning algorithms that by default assume the input space is vectorial—whether for supervised classification, feature selection, or unsupervised image clustering. Csurka *et al.* [CBDF04] first showed this connection for recognition by using the bag-of-words descriptors for discriminative categorization. Since then, many supervised methods exploit the bag-of-words histogram as a simple but effective representation. In fact, many of the most accurate results in recent object recognition challenges

employ this representation in some form [EVGW+, Cal04].

Most recently, a number of methods for unsupervised topic modeling have been built upon bag-of-words features [SRE+05, RES+06, FFFPZ05, QMO+05], once again taking inspiration from methods originally used in document/text processing. For example, Russell and colleagues explore how probabilistic Latent Semantic Analysis and Latent Dirichlet Allocation can be used to discover the visual themes among segments extracted from unlabeled images. While there is no guarantee of discovering patterns at the object level, their results demonstrate that object categories do tend to surface among the repeated elements.

The lack of geometry in the bag-of-words representation (BoW) can potentially be either an advantage or a disadvantage. On the one hand, by encoding only the occurrence of the appearance of the local patches, not their relative geometry, we get significant flexibility to viewpoint and pose changes. On the other hand, the geometry between features can itself be an important discriminating factor, which a BoW will miss. Assuming none of the patches in an image overlap, one would get the same description from a BoW no matter where in the image the patches occurred. In practice, features are often extracted such that there is overlap, which at least provides some implicit geometric dependencies among the descriptors. Furthermore, by incorporating a post-processing spatial verification step, or by expanding the purely local words into neighborhoods and configurations of words, one can achieve an intermediate representation of the relative geometry. We discuss this in more detail in Chapter 6, Section 6.2.

When the BoW is extracted from the whole image, features arising from the true foreground and those from the background are mixed together, which can be problematic, as the background features "pollute" the object's real appearance. To mitigate this aspect, one can form a single bag from each of an image's segmented regions (possibly from multiple segmentations), or in the case of sliding window classification, within a candidate bounding box sub-window of the image.

Finally, in spite of clear benefits that visual words afford as tools for recognition, the optimal formation of a visual vocabulary remains unclear. The analogies drawn between textual and visual content only go so far: real words are discrete and human-defined constructs, but the visual world is continuous and yields complex natural images. Real sentences have a one-dimensional structure, while images are 2D projections of the 3D world. Thus, more research is needed to better understand the choices made when constructing vocabularies for local features.

In the next chapter we will discuss an alternative view for recognition with local features, where instead of summarizing an image's distribution, one seeks a correspondence or matching between the candidate parts.