Efficient Object Discovery Based on Locality Sensitive Hashing

by

Gibran Fuentes Pineda

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Engineering

at the

Graduate School of Information Systems

The University of Electro-Communications
Tokyo, Japan

September 2011

Efficient Object Discovery Based on Locality Sensitive Hashing

by

Gibran Fuentes Pineda

To Mariana

Abstract

In most object recognition methods, the creation of object models requires human supervision such as manual object segmentation, image annotations or the number of different kinds of objects that appear in a given set of images. However, for large image sets, even a small amount of supervision can be extremely expensive. Moreover, the performance of these methods deteriorates with the number of images and/or the dimensionality of the image representation. These limitations make current methods unsuitable for exploiting the large amount of visual information available nowadays.

This dissertation addresses the problem of efficient object discovery from images without supervision. Given a set of images, the goal is to automatically derive models that can be used to retrieve images that contain particular objects and to localize such objects within the images. We are interested in finding associations between images based on the objects they contain rather than on the whole image. We propose a new approach to discovering objects by searching for reoccurring patterns via hashing. In particular, we exploit locality sensitive hashing (LSH), a randomized algorithm for efficient similarity search. Because of the consistent use of hashing, such approach is highly efficient and suitable for large image sets. We develop two object discovery methods based on this approach.

In our first method, region features are computed from the images by color segmentation. Then, objects are discovered by extracting frequent patterns of closely located regions. Here, we exploit hashing to (1) extract components by gathering near pixels of the same color, (2) label similar components based on their color and size, (3) generate object candidates by gathering closely located components and (4) find similar object candidates. This method can discover objects in simple scenes robustly against rotation, slide and small intra-class variations.

Our second method represents each image as a set of vector quantized affine covariant features. Object models are extracted by clustering features that consistently appear together in the same images under the assumption that they underlie the same object. The extraction of co-occurring features is efficiently implemented by Min-Hashing. The models derived by this method are highly discriminative and robust to clutter, occlusion and changes in scale, illumination and viewpoint. The proposed method could discover objects from a set of 101,922 images in just 38.35 minutes. In a quantitative evaluation using a benchmark dataset, this method had higher scores than state-of-the-art methods while being significantly faster.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

One of the critical problems in computer vision is to understand the content of images. This problem has become more important in recent years due to the rapidly increasing amount of visual information available. Today, digital cameras are ubiquitous and building a personal photo collection is a widespread practice. Moreover, commercial catalogs archives, stock photography agencies and photo-sharing websites store huge image sets which are accessible through the Internet. Notably, several Internet websites provide free photo storage to millions of users. Table 1.1 presents the number of images currently stored by some of these Internet websites. Such large image sets represent a very rich source of visual information which have been recently exploited by computer vision researchers to leverage applications such as automatic 3D reconstruction of buildings [7, 52, 53], landmark discovery [45, 54, 64] and landmark recognition [36].

Table 1.1: Internet websites with huge image sets.

Website	**# of Images**	**URL**
Flickr [1]	5 Billion	`http://www.flickr.com/`
Picasa [2]	7 Billion	`https://www.picasaweb.google.com/`
Photobucket [3]	8 Billion	`http://photobucket.com/`
Facebook [2]	60 Billion	`http://www.facebook.com/`

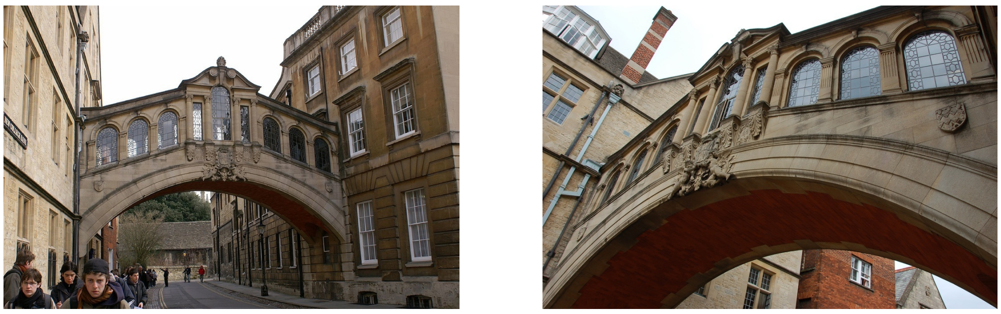


Figure 1.1: An object seen from two very different viewpoints.

Typically, organizing and searching image sets is realized through manually generated annotations. However, these annotations do not always provide an accurate description of the image content. Therefore, extracting information about the visual content of the images has become necessary. In many cases, what is required is to know the kinds of objects that are present in the images. This information can be used to present a visual summary of the image set, improve the efficiency and accuracy of image/object retrieval, or classify the images based on the kinds of objects they contain.

Recognizing objects from images has been a long standing problem in computer vision and although significant progress has been made, there are still many challenges that need to be addressed. In order to be able to recognize objects, these should be modeled by some means so that computers can handle them. In most object recognition methods such models must be created under some kind of human supervision, which may range from manually segmenting the object within the image or labeling the image with the kinds of objects it contains to just providing the number of different kinds of objects that appear in the image set. However, for extremely large image sets with highly diverse contents, even a small amount of supervision can be extremely laborious. This issue becomes worse if one considers the high time and space complexity of many object recognition methods. Typically, they just can handle small image sets and a limited number of object kinds because their performance is badly affected by the high dimensionality of the images. In addition, the recognition is usually based on the whole image which results in a poor performance for scenes with large amounts of clutter and partial occlusions.

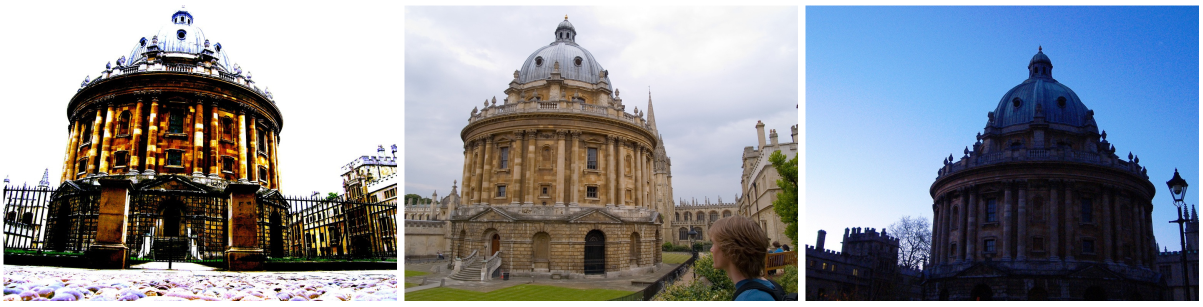


Figure 1.2: Example of an object with variations in appearance due to changes in illumination.

1.2 Objectives

The main objective of this research is to develop methods for discovering objects from unannotated images without supervision. Given a set of images, the aim is to derive models that can be used to retrieve images that contain particular objects and to localize such objects within the images. We are interested in finding associations between images based on the objects they contain rather than on the whole image. In particular, we are concerned with methods that can be applied to large sets of complex natural scenes such as those found in photo-sharing websites. For object discovery to be successful in such image sets, the following problems must be addressed:

- **Scale changes.** Objects may appear within the images at a wide range of scales. Invariance to scale changes is therefore a desirable characteristic of object discovery.
- **Viewpoint changes.** The same object seen from different viewpoints can look very different. Figure 1.1 illustrates an example of the variation in the object appearance due to changes in viewpoint.
- **Illumination changes.** The object appearance can also vary due to changes in illumination. Images of the same object with very different illumination are shown in Fig. 1.2.
- **Large amounts of clutter.** As shown in Fig. 1.3, natural scenes often contain multiple objects and other backgrounds. In such complex scenes, it can be extremely difficult to discriminate the background from the objects of interest and in many cases, the object

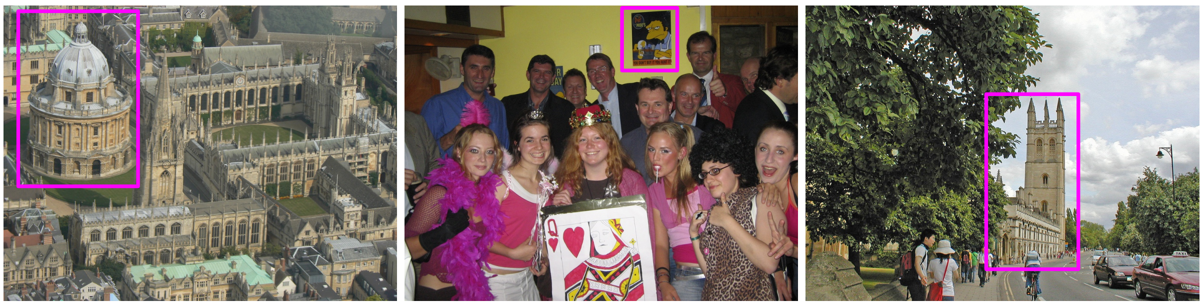


Figure 1.3: Examples of objects in scenes with large amounts of background clutter. The bounding box contains the object of interest within the image.

represents only a small part of the whole image. In this research, we seek to discover objects despite large amounts of background clutter.

- **Occlusions**. An object may appear occluded by other objects (see Fig. 1.4), leaving it only partially visible. A successful object discovery method must have the ability to deal with partial occlusions.

- **Cardinality of the image set.** Since we are interested in dealing with large numbers of images, efficient mechanisms to index, discover and retrieve objects are of paramount importance. In this research, we pay special attention to the efficiency and scalability of the object discovery.

- **Diversity of object kinds.** An image set may include a wide variety of different objects. Ideally, an object discovery method should be able to derive a model for any kind of object and at the same time, the derived object models must have good discrimination power.

1.3 Contributions

The most important contributions of this research are two data-driven methods for extracting object models from unordered image sets without supervision. These methods are designed based on the next three key ideas.

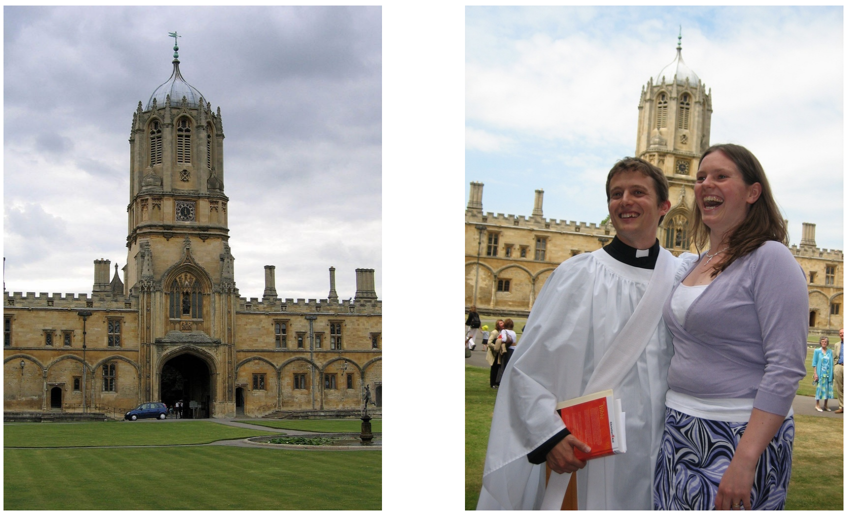


Figure 1.4: A partially occluded object: clear view of the object (left) and the object occluded by other objects (right).

1. An object is represented by a combination of components. We adopt a component-based representation because of its flexibility and robustness to clutter, occlusion and variations in appearance.

2. To extract meaningful objects without supervision, we pay attention to reoccurring patterns of components in the image set. Searching for reoccurring patterns is one of the most successful strategies in data mining for extracting meaningful and interesting information from structured (e.g. a transaction database) and semi-structured (e.g. text) data. Here, we demonstrate that a reoccurring pattern in an image set often underlies a meaningful object.

3. To achieve scalability, we implement the extraction of reoccurring patterns efficiently by relying on locality sensitive hashing (LSH), a randomized algorithm for efficient similarity search which has proved to be particularly suitable for handling large sets of high dimensional data (see [16, 17, 27]).

In Table 1.2, we present the characteristics of our two object discovery methods. In the following, we summarize these two methods.

Table 1.2: Summary of the proposed methods.

Proposed Method	**Object Components**	**Reoccurring Patterns**	**Hashing Scheme**	**Robustness**
Region-based	Image regions (by color segmentation)	Frequent closely located regions	Euclidean LSH + Standard Hashing	Slide, in-plane rotation, small intra-class
Feature-based	Visual words (vector quantized affine covariant features)	Co-occurring visual words	Min-Hashing	Slide, in-plane rotation, occlusion, clutter, scale, illumination and viewpoint

Region-based object discovery We first develop a method for discovering objects from color segmented images. Here, each image region corresponds to a single object component. Image regions are classified according to their attributes (e.g. color) and the background regions are removed so that objects are isolated. Then, assuming that objects do not overlap each other, the method searches for patterns of closely located regions and considers one frequent pattern as a meaningful object. By searching the known objects, the same strategy can be used for recognizing an unknown object. We implement this method completely by hashing. Specifically, we employ hashing to efficiently realize the next three kinds of similarity judgments: (1) standard distance-based similarity judgment, (2) distance-based similarity judgment considering the relative size (e.g. large regions are considered as similar with greater differences between their sizes than small regions), and (3) matching robust to small variations. This method can efficiently discover objects against simple background clutter, rotation, slide and small intra-class variations.

Feature-based object discovery Since our first method assumes that objects are isolated from the background and they do not overlap each other, it can not handle large amounts of background clutter or partial occlusions. To overcome this problem, we propose another method which pays attention to components that co-occur consistently in the same images. The rationale is that components that belong to the same object tend to appear together much more often than those belonging to different objects. Hence, our method yields object models by clustering components that consistently appear together in the image set. Remarkably, we accelerate the extraction of co-occurring components by Min-Hashing. In addition, as current color segmentation methods produce different results even with slightly different images, in this method we use vector quantized affine covariant features which are stable under large image variations. The models derived by our feature-based method are highly discriminative and robust to clutter, occlusion and changes in scale, illumination and viewpoint. This method could discover objects that correspond to human annotated objects from a set of 101,922 images in just 38.35 minutes. In a quantitative evaluation using ground truth data, the proposed method outperformed state-of-the-art methods both in accuracy and efficiency.

1.4 Organization of the Dissertation

The remainder of this dissertation is organized as follows:

- **Chapter 2** provides the background knowledge necessary to understand this dissertation. We review previous works on object recognition with a special focus on methods for the recognition of particular objects. This chapter also contains a brief introduction to locality sensitive hashing (LSH). In particular, it describes Euclidean LSH and Min-Hashing, which are essential for the implementation of our two object discovery methods.

- **Chapter 3** introduces our region-based object discovery method. In this chapter, we describe an approach to finding frequent patterns of closely located image regions. We conduct a series of experiments to evaluate the validity of the discovered objects. Finally, we point out the strengths and weaknesses of this method.

- **Chapter 4** introduces our feature-based object discovery method. Here, each image is represented by a set of vector quantized affine covariant features. We develop, under this framework, a scalable approach supported by Min-Hashing to clustering co-occurring features from an image set. Qualitative and quantitative evaluations using a benchmark dataset are performed to show the meaningfulness and robustness of the derived object models. We also analize the scalability of the object discovery method. At the end of the chapter, we discuss the main characteristics of this method.

- **Chapter 5** gives the concluding remarks of the dissertation and discusses possible future directions.

Chapter 2

Background

In this chapter, we present the background knowledge necessary to better understand the rest of this dissertation. We begin with a literature review on object recognition technologies. Special attention is paid to technologies for recognizing particular objects rather than object categories. Then, the second part of the chapter provides a brief introduction to locality sensitive hashing (LSH). In particular, it describes Euclidean LSH and Min-hashing, which are essential for the implementation of our two object discovery methods.

2.1 Object Recognition

Object recognition from images has been challenging problems in image analysis over the past decades and yet there is no complete and integrated solution to this problem. In general, object recognition can be classified into two different problems: the recognition of object categories such as motorbikes or human faces (see Fig. 2.1) and the recognition of particular objects such as an specific magazine or building (see Fig. 2.2). In this research, we focus on the recognition of particular objects.

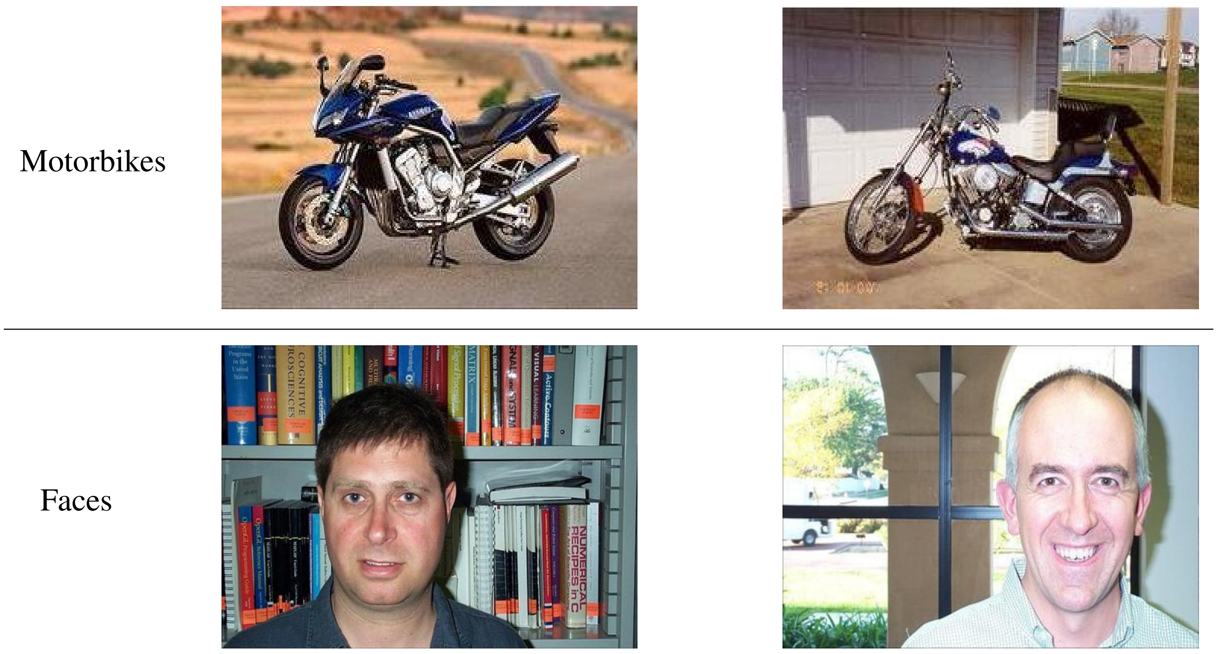


Figure 2.1: Examples of object categories.

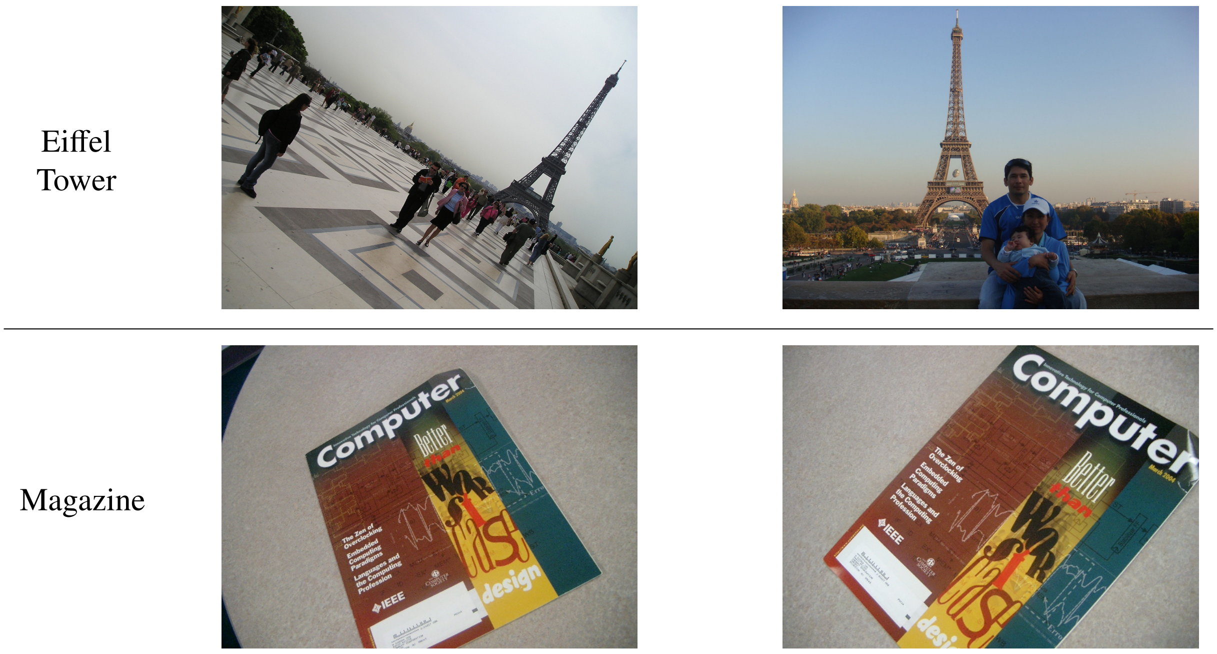


Figure 2.2: Examples of particular objects.

2.1.1 Traditional Approaches

The primary goal of object recognition is to construct a model that describes an object under varying imaging conditions. Object recognition involves two essential tasks, namely modeling and matching (or recognition). Figure 2.3 shows the overview of object recognition.

To contruct the object models, we need a suitable representation. In general, approaches of object representation can be divided into three:

- **Model-based.** In these approaches, objects are represented by a set of three dimensional primitive geometric elements such as cones, spheres and planes.
- **Shape-based.** These approaches use the contour information of the objects to represent them.
- **Appearance-based.** The distribution of the pixel intensities is exploited to represent the objects. Appearance-based representations can further be based on global or local features. Global approaches usually project the whole image onto a simpler low-dimensional vector space by means of principal component analysis [55, 59], Gabor filters [33–35], Wavelets [46, 57] and so on, while local approaches represent an object by combining local image features (e.g. [11,20,23,24,41,50,60]).

Modeling an object class is realized by either constructing an approximate representation (generative models) or defining an optimal decision boundary (discriminative models) from a set of given examples. On the other hand, to match a query object to the most similar class, generative models compute the similarity between the query object and each model whereas discriminative models predict the class from the estimated decision boundary.

Several methods have been proposed using many different algorithms for representing and modeling objects. Although many of these methods have shown good performance for some applications, they usually scale poorly to very large image sets with highly diverse contents because:

- They need to manually specify the set of examples that correspond to each class.

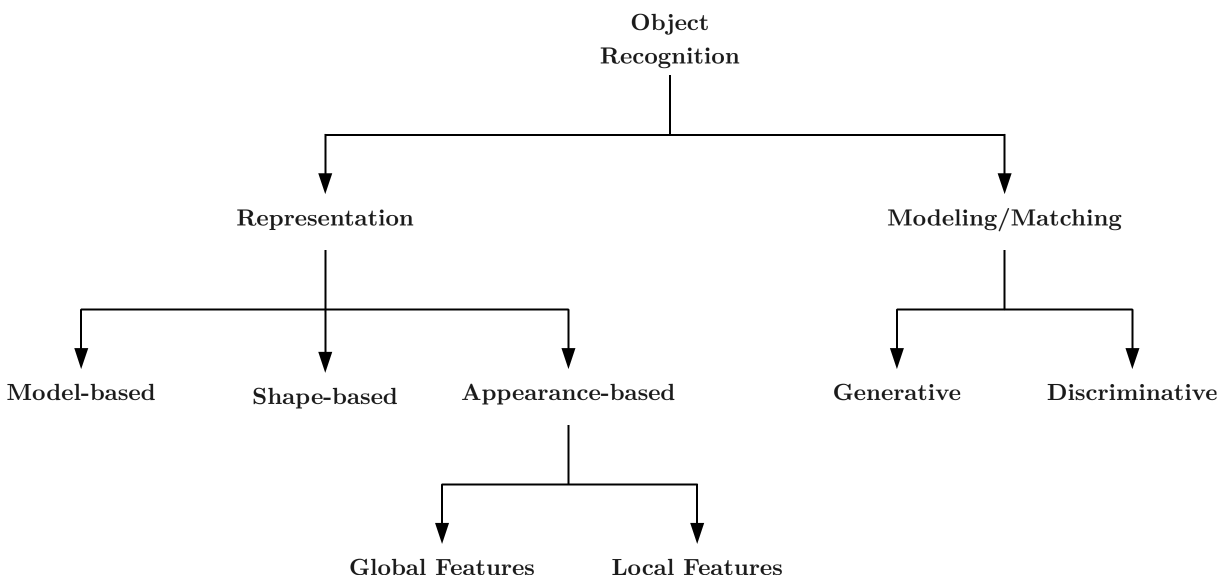


Figure 2.3: Overview of object recognition.

- Their time complexity greatly increases with the dimensionality of the representation and the number of classes.
- They are tailored to specific classes (e.g. faces or cars).

2.1.2 Scalable Approaches

Modern feature detectors and descriptors have boosted the development of efficient techniques to represent large collections of images and videos. In particular, the bag-of-features (BOF) approach [51], first introduced by Sivic et al. [51] to retrieving frames from a video, has been widely adopted. In [51], shape adapted regions around Harris interest points as well as maximally stable extremal regions are detected in all the frames of a video. Each of the detected regions is described by a SIFT vector. Then, [51] clusters the SIFT vectors by using k-means where a cluster center corresponds to a visual word. Thus, each SIFT vector is assigned to the nearest visual word. In this way, [51] represents an image as a set of visual words. This approach has been widely adopted due to its simplicity, flexibility and excellent performance. Furthermore, a BOF model is robust to occlusion, clutter and changes in scale, illumination and viewpoint. In the following, we review some object discovery methods that are based on the BOF approach.

State-of-the-art image retrieval systems can efficiently retrieve images of the same object even in large image collections [32, 40, 42] by relying on the BOF approach. However, most of these systems only compute global similarity between images by counting the number of shared visual words. Therefore, their ability to recognize the same objects is limited, especially when the objects do not cover the entire image in a complete form.

Chum and Matas [16] proposed a fast algorithm for discovering related images based on an extension of Min-Hashing [18]. This algorithm hashes images to find similar image pairs and then forms clusters of spatially related images. It is not hard to see that as the number of common visual words between two images decreases, they are unlikely to be treated as similar. This is a disadvantage that limits the ability to cluster images of the same object, especially when the object occupies a small portion of an image.

Motivated by the success of topic discovery from documents, many researches have relied on latent variable models such as PLSA [28] and LDA [13] to discover objects from images [49, 56, 61]. Latent variable models represent each image as a mixture of K topics where each topic corresponds to a single object class. One important limitation of these methods is that the number of topics K must be given a priori. Even slightly different choices of K might lead to quite different results. This limitation becomes worse when the image collection is large and diverse because the number of topics can be hard to infer. Furthermore, as it is very time consuming to estimate the model parameters, latent variable models are not easily scalable to large databases.

Philbin et al. [43, 44] mine objects from large image collections. Both of [43] and [44] first use image retrieval techniques to build a matching graph which divides the image collection into groups of spatially related images. Then, [43] performs spectral clustering to partition the groups that contain multiple disjoint objects, whereas [44] employs gLDA (a variant of LDA that takes into account geometric information) on each group to generate object models. An important drawback of these methods is that the construction of the matching graph is very expensive. In addition, applying spectral clustering or gLDA to each group of the matching graph is also very time consuming, especially when there are large groups.

2.2 Similarity Judgments via Hashing

Similarity judgment is a fundamental element for pattern recognition in image analysis systems. These systems typically model similarity (or more properly dissimilarity) with a metric. In many image analysis tasks items are defined by many attributes and multiple similarity measures might be necessary. For example, a set of objects can be characterized by their components and shape. For the former, a set-theoretic similarity measure is adequate while the Euclidean distance is suitable for the latter. However, as most image analysis schemes presume only specific spaces and similarity measures (commonly the Euclidean distance), it is not guaranteed that the same scheme will have the same good performance when applied to other spaces and/or similarity measures. The complexity of the system increases if one expects to support several schemes simultaneously to treat different kinds of similarity measures. Furthermore, in very large sets of high dimensional data, the time complexity due to the "curse of dimensionality" imposes an important difficulty.

Since hashing techniques provide an efficient searching mechanism for various similarity judgments that are common in image analysis tasks, we believe that it is possible to construct a simple and efficient image analysis system by using such techniques. Hence, we consistently rely on hashing techniques inspired by the *locality-sensitive hashing* (LSH) [29] scheme of Indyk and Montwani. LSH inherits the constant time and ease of implementation from standard hashing while it can map similar items from a given space to the same bucket in the bucket space. This is in contrast to standard hashing, where non-identical items are mapped to different buckets in the bucket space, even if they are very similar. Figure 2.4 shows the differences of the mapping properties between standard hashing and LSH. In this section, we describe in detail LSH and mention several other techniques to judge similarity by hashing.

2.2.1 Locality Sensitive Hashing (LSH)

LSH performs approximate similarity search in high dimensional spaces. The basic idea is to project the high dimensional space $\mathbb{X}^d$ to a low dimensional subspace in such a way that the

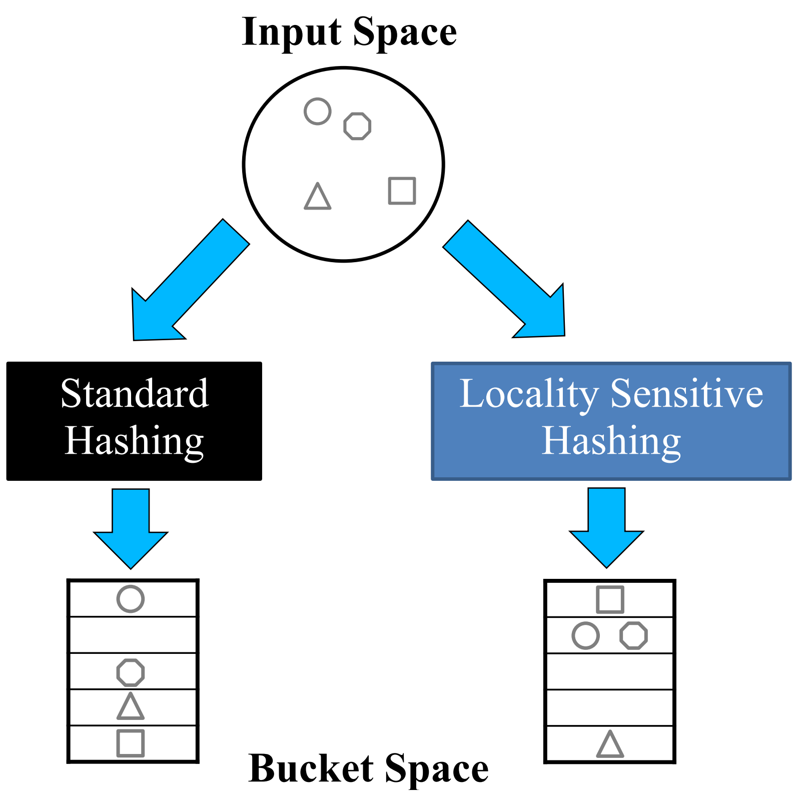


Figure 2.4: Comparison between standard hashing and LSH mapping properties.

distances between all points in $\mathbb{X}^d$ are approximately preserved in the projected space with high probability. To achieve this, LSH defines a family of hash functions $\mathcal{H}$ (called *locality-sensitive* for a dissimilarity measure D) satisfying the following definition.

Definition 1 *A family of hash functions* $\mathcal{H} = h : \mathbb{X}^d \rightarrow \mathbb{U}$ *is called locality-sensitive for* D *if for any* $X_i, X_j \in \mathbb{X}^d$*, there exist real numbers* r_1, r_2, p_1, p_2 *such that the next two properties hold.*

- $D(X_i, X_j) \leq r_1 \Rightarrow P[h(X_i) = h(X_j)] \geq p_1,$

- $D(X_i, X_j) \geq r_2 \Rightarrow P[h(X_i) = h(X_j)] \leq p_2,$

where $P[E]$ denotes the probability of the event E.

Generally, it is desirable that $p_1 > p_2$ and $r_1 < r_2$ when D is a distance function. For most applications, the gap between the two probabilities p_1 and p_2 must be amplified. This is accomplished by introducing tuples $g_1, g_2, \ldots g_l$ of k hash functions ($k \gg 1$) selected independently and uniformly at random from $\mathcal{H}$, i.e.,

$$\begin{array}{rcl} g_1 & = & (h_{11}, \quad h_{12}, \quad \ldots, \quad h_{1k}) \\ g_2 & = & (h_{21}, \quad h_{22}, \quad \ldots, \quad h_{2k}) \\ & \vdots & \\ g_l & = & (h_{l1}, \quad h_{l2}, \quad \ldots, \quad h_{lk}) \end{array} . \tag{2.1}$$

Thus, LSH builds l hash tables (one for each g) and stores each point in $\mathbb{X}^d$ at each of these tables. This data structure allows to perform approximate similarity search efficiently even in high dimensional spaces.

Several LSH families have been proposed for different spaces and similarity measures. Some of the similarity measures for which LSH families have been discovered include the Euclidean distance [26], Hamming distance [29], ℓ_s-distance [21], Jaccard coefficient [14, 19], Manhattan distance [8, 37] and Earth Mover's distance [15]. The reader is referred to [9] for further information on LSH families.

Euclidean LSH

A simple LSH scheme for Euclidean spaces that achieves a time complexity linear in d and sublinear in the number of data points was proposed by Gionis et al. [26]. We describe this scheme hereinafter. Let P be a set of points in a d-dimensional space and C be the maximum coordinate value of any point in P. Every $p \in P$ is transformed to a Cd-dimensional vector by concatenating unary expressions for every coordinate, that is,

$$f(p) = \text{Unary}(x_1)\text{Unary}(x_2)\cdots\text{Unary}(x_d), \tag{2.2}$$

where Unary(x) is a sequence of x ones followed by $C - x$ zeros. In [26], a tuple $g(p)$ of hash functions from a point p is computed by picking up k bits independently and uniformly at random from these Cd bits and concatenating them (each of these selected bits corresponds to a different hash function h). This process can be seen as a partitioning of the d-dimensional space into cells of different sizes by a set of k random hyperplanes. The intuition is that near points will lie in the same cell with high probability. Figure 2.5 (a) illustrates an example of the

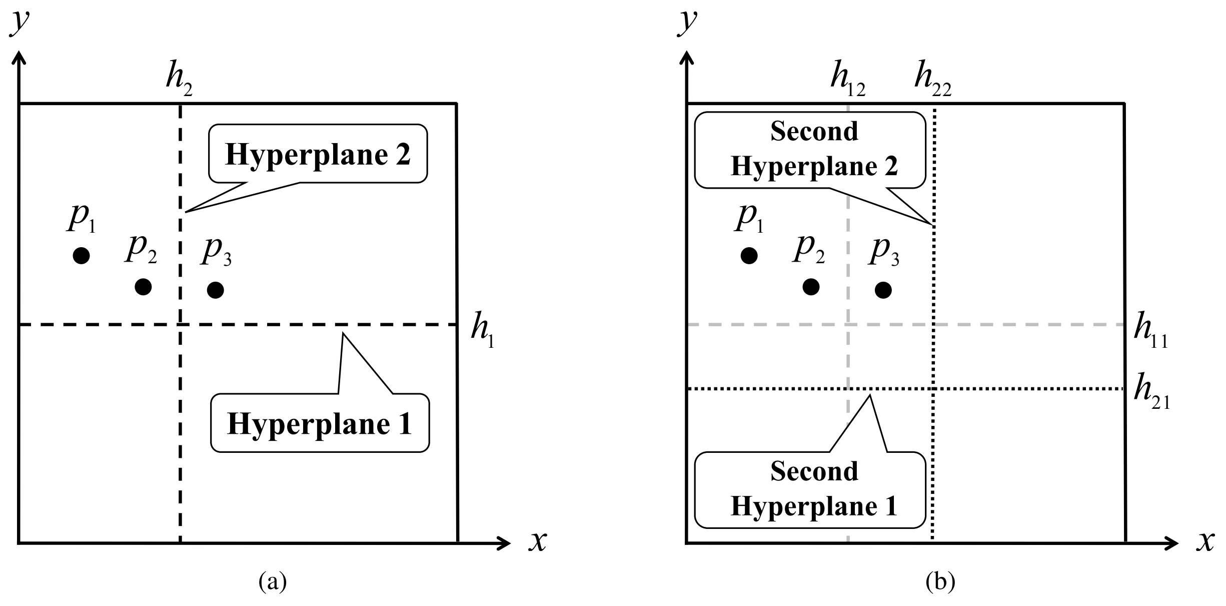
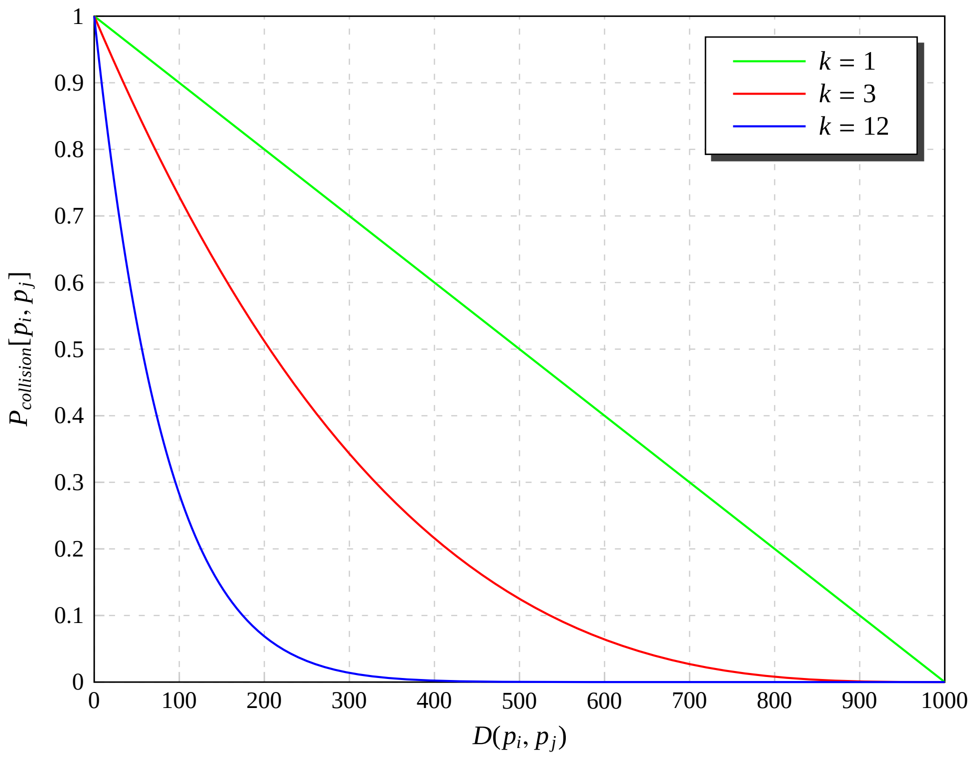


Figure 2.5: Space partitioning by Euclidean LSH.

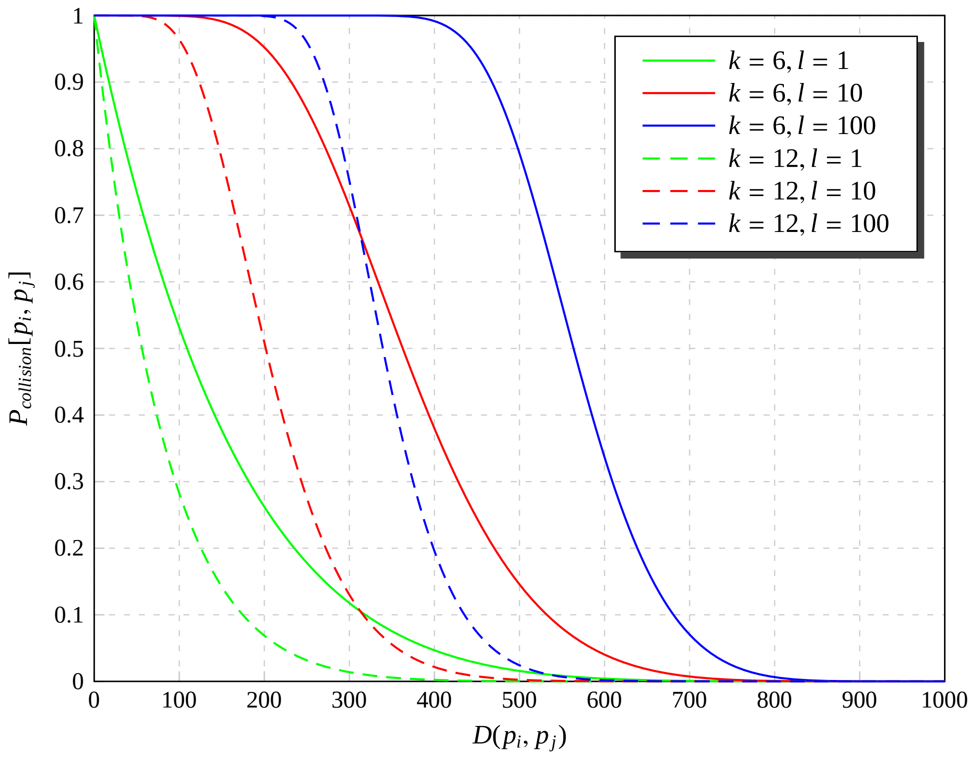
space partitioning by a set of random hyperplanes. In this example, a 2-dimensional space is partitioned into 4 cells by a set of 2 hyperplanes (hash functions h_1 and h_2, where $k = 2$). Note that the near points p_1 and p_2 lie in the same cell whereas p_1 and p_3 are separated into different cells because they are far from each other. Depending on the space partitioning, near points are likely to be separated into different cells. For example in Fig. 2.5 (a), p_2 and p_3 are close to each other but they lie in different cells. To exclude this failure, multiple l sets of random hyperplanes (tuples $g_1, g_2, \cdots g_l$) are prepared. Figure 2.5 (b) shows that by partitioning the space with two different sets of random hyperplanes $g_1 = (h_{11}, h_{12})$ and $g_2 = (h_{21}, h_{22})$, p_2 and p_3 lie in the same cell.

In general, as k becomes large, remote points are less likely to lie in the same cell because the sizes of the generated cells become smaller. In Fig. 2.6 (a), we present the probability for different values of k that two points lie in the same cell as the distance between them increases. On the other hand, by defining multiple l tuples, near points will lie in the same cell at least in one tuple with high probability. Figure 2.6 (b) presents the probability that two points lie in the same cell in at least one tuple for different values of k and l.

The implementation of this LSH scheme is achieved by constructing a hash table for each tuple (l different hash tables in total), where buckets correspond to cells and points that lie in



(a)



(b)

Figure 2.6: Probability that two points lie in the same cell as a function of their distance for: (a) different values of k ($l = 1$) and (b) different values of k and l.

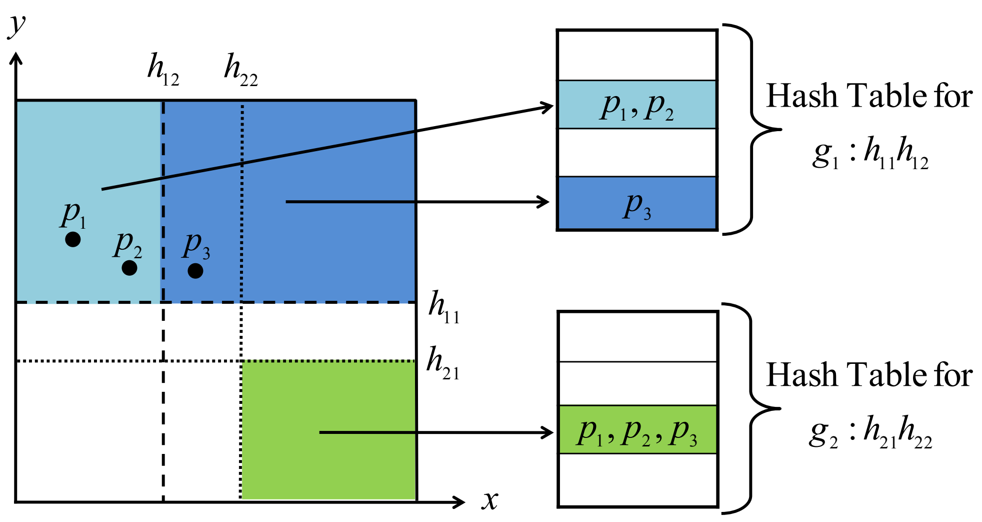


Figure 2.7: Retrieval of near points by Euclidean LSH.

the same cell are stored in the same bucket (see Fig. 2.7).

Min-Hashing

Min-Hashing [14] is a randomized algorithm for efficiently computing the *Jaccard similarity* between sets. In this section, we give a brief overview of Min-Hashing. For a more detailed explanation, the reader is referred to [14] (see also [19]).

Let X_i and X_j be a pair of sets whose elements are chosen from M different items $\{1, 2, \ldots, M\}$. The Jaccard similarity between X_i and X_j is defined as

$$sim(X_i, X_j) = \frac{| X_i \cap X_j |}{| X_i \cup X_j |} \in [0, 1]. \tag{2.3}$$

In Min-Hashing, we select a random permutation π of the ordered items $\{1, 2, \ldots, M\}$. From the viewpoint of combinatorics, since the number of different items is M, $M!$ permutations of the items are possible. Here, the permutation π is selected at random from these $M!$ different permutations. After π is determined, the min-hash value for X_i is computed as its first element after X_i is permuted according to π. That is,

$$h(X_i) = min(\pi(X_i)), \tag{2.4}$$

where $\pi(X_i)$ denotes the permutation of X_i under π. For example, let $\pi = \{2, 5, 4, 3, 1\}$ be a random permutation of the ordered items $\{1, 2, 3, 4, 5\}$. Now consider two sets $X_1 = \{1, 2, 3\}$ and $X_2 = \{1, 3, 4\}$. The first element of X_1 on π is 2 whereas the first element of X_2 is 4. Therefore, $h(X_1) = 2$ and $h(X_2) = 4$. In practice, the selection of a random permutation is implemented by assigning a random number to each item. Then, the min-hash value of a set is obtained by finding the minimum of the numbers assigned to its elements.

In Min-Hashing, the probability that X_i and X_j take the same min-hash value is equal to their Jaccard similarity [19]. Namely

$$P[h(X_i) = h(X_j)] = sim(X_i, X_j). \tag{2.5}$$

This is because two sets X_i, X_j are assigned the same min-hash value only when the first element of both X_i and X_j on π is the same, that is, $h(X_i) = h(X_j) \in X_i \cap X_j$. Since all the elements have the same probability of being the first on π, the probability of $h(X_i) = h(X_j)$ is equal to the number of elements that X_i and X_j have in common over the number of elements that are either in X_i or X_j. In the above example, the probability that X_1 and X_2 take the same min-hash value is $2/3$.

Hence, similar sets will have the same min-hash value with high probability. However, because Min-hashing is a probabilistic method, false negatives (similar sets with different min-hash values) and false positives (dissimilar sets with the same min-hash value) are likely to happen. To overcome this problem, multiple min-hash values are computed to judge whether two sets are similar or not, where each min-hash value is obtained under a different permutation selected independently at random from the $M!$ permutations. Following the previous example, we compute 4 min hash values h_1, h_2, h_3, h_4 for X_1 and X_2 using the next 4 different random permutations:

$$\begin{array}{l} \pi_1 = \{2, 5, 4, 3, 1\} \longrightarrow (h_1(X_1) = 2, h_1(X_2) = 4) \\ \pi_2 = \{5, 3, 1, 4, 2\} \longrightarrow (h_2(X_1) = 3, h_2(X_2) = 3) \\ \pi_3 = \{3, 1, 4, 2, 5\} \longrightarrow (h_3(X_1) = 3, h_3(X_2) = 3) \\ \pi_4 = \{3, 4, 1, 5, 2\} \longrightarrow (h_4(X_1) = 3, h_4(X_2) = 3) \end{array} . \tag{2.6}$$

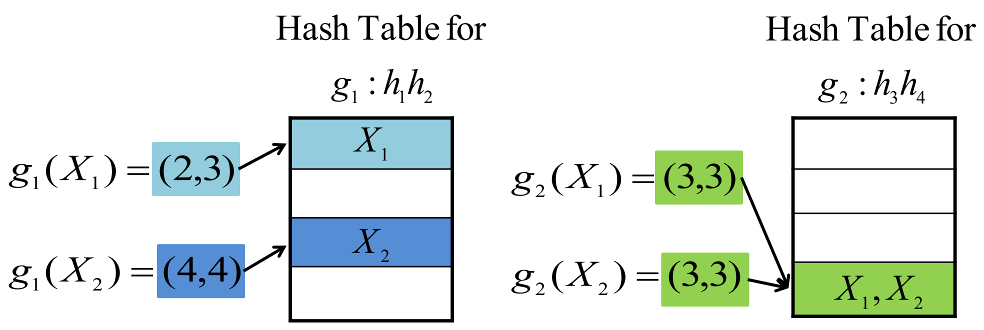


Figure 2.8: Retrieval of similar sets by Min-Hashing.

Retrieving similar sets is achieved by grouping the min-hash values into l tuples $g_1, \ldots, g_l$, each of them composed of r different min-hash values. The l tuples for a set X_i are defined as follows:

$$\begin{aligned} g_1(X_i) &= (h_1(X_i), h_2(X_i), \ldots, h_r(X_i)) \\ g_2(X_i) &= (h_{r+1}(X_i), h_{r+2}(X_i), \ldots, h_{2 \cdot r}(X_i)) \\ &\ldots \\ g_l(X_i) &= (h_{(l-1)\cdot r+1}(X_i), h_{(l-1)\cdot r+2}(X_i), \ldots, h_{l \cdot r}(X_i)) \end{aligned} \quad . \tag{2.7}$$

Here $h_j(X_i)$ denotes the j-th min-hash value. Note that $r \cdot l$ min-hash values are used in total, as r min-hash values are necessary for each tuple g_i $(1 \leq i \leq l)$. Thus, l hash tables are constructed (one for each tuple), and two sets X_i, X_j are stored in the same hash bucket on the k-th hash table, if $g_k(X_i) = g_k(X_j)$ (see Fig. 2.8).

As very similar sets are expected to agree in several min-hash values, they will be stored in the same hash bucket with high probability. In contrast, dissimilar sets will seldom have the same min-hash value and therefore the probability that they collide will be low. More precisely, the probability that two sets X_i, X_j agree in the r min-hash values of a given tuple g_k is

$$P[g_k(X_i) = g_k(X_j)] = sim(X_i, X_j)^r, \tag{2.8}$$

because all the r min-hash values of g_k have to be the same. Consequently, the probability that two sets X_i, X_j have at least one identical tuple (i.e. they are stored in the same bucket in at least one hash table) becomes

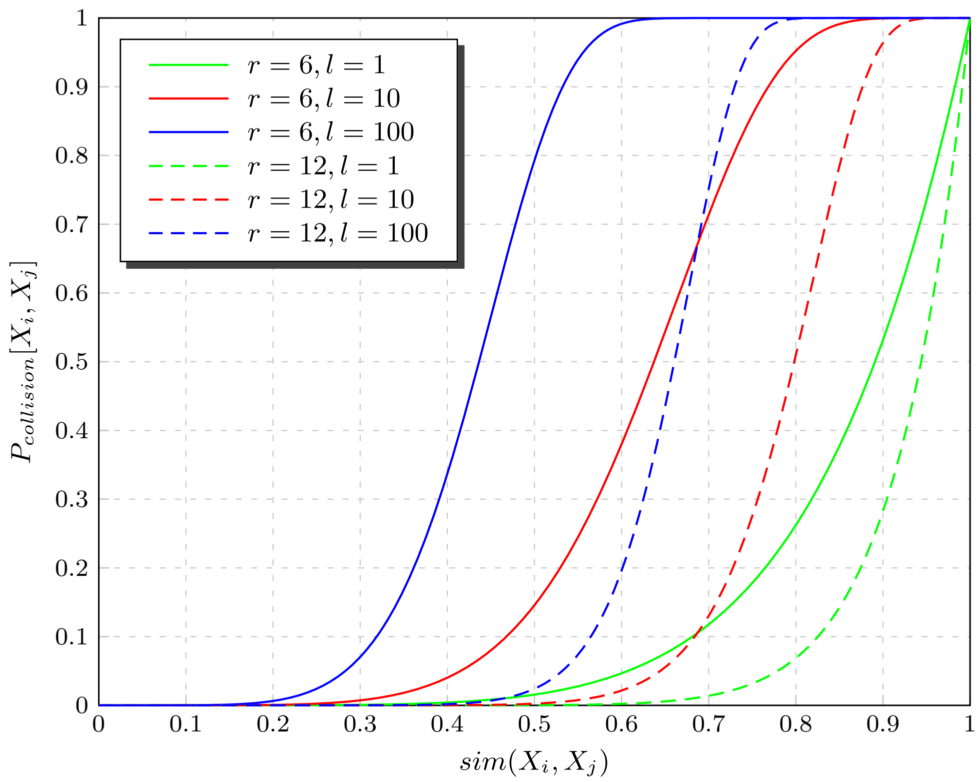


Figure 2.9: Probability of collision of two sets as a function of their similarity for different values of *r* and *l*.

$$P_{collision}[X_i, X_j] = 1 - (1 - sim(X_i, X_j)^r)^l. \tag{2.9}$$

In Fig. 2.9, we present the graph of $P_{collision}[X_i, X_j]$ as a function of the similarity between X_i and X_j for different values of *r* and *l*. By choosing *r* and *l* properly, this probability approximates a unit step function such that

$$P_{collision}[X_i, X_j] \approx \begin{cases} 1, & \text{if } sim(X_i, X_j) \geq s* \\ 0, & \text{if } sim(X_i, X_j) < s* \end{cases}, \tag{2.10}$$

where $s*$ is a threshold parameter. That is to say, the probability of collision for sets whose similarity is greater than s* is close to 1 whereas for sets with similarity lower than s* is close to 0. Here, the selection of *r* and *l* is a trade-off between recall and precision. Figure 2.10 shows the graph of $P_{collision}[X_i, X_j]$ for $s* = 0.6$ with different selections of *r* and *l*.

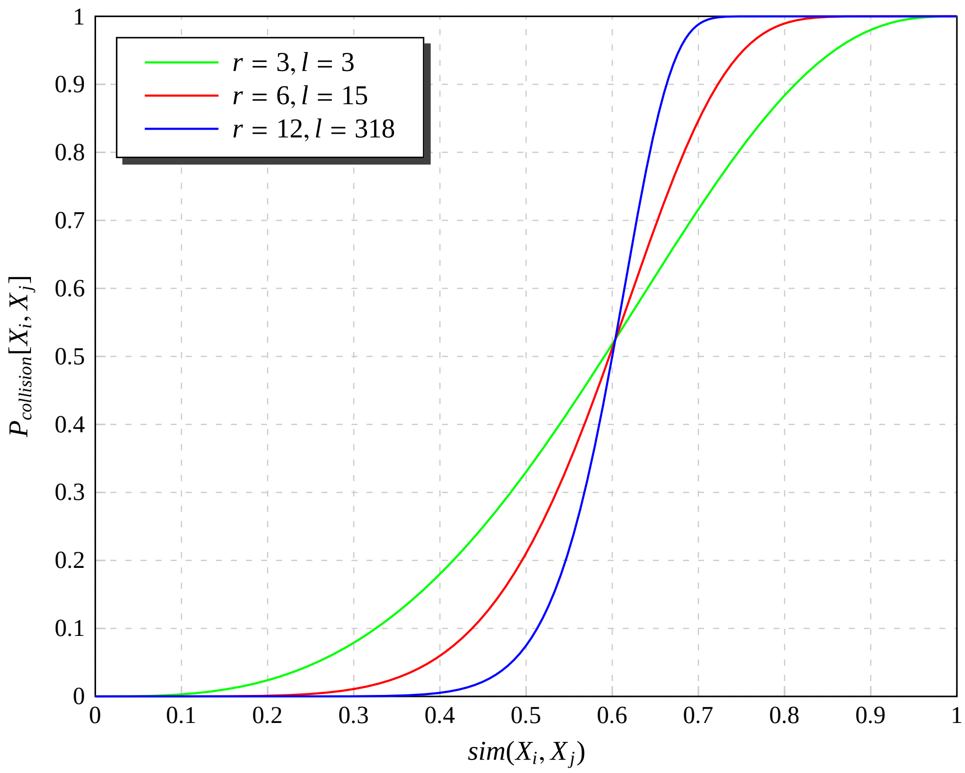


Figure 2.10: Probability of collision of two sets for different selections of r and l ($s* = 0.6$).

Given $s*$ and r, we can determine l by setting $P_{collision}[X_i, X_j]$ to 0.5, which gives

$$l = \frac{log(0.5)}{log(1 - S*^r)}. \tag{2.11}$$

In this way, we can use Min-Hashing to retrieve those sets whose similarity is greater than a threshold.

2.2.2 Other Hashing Techniques

Although the LSH scheme is highly efficient, there exist some spaces in which LSH families have not yet been identified and therefore LSH is not applicable. In such cases, other hashing techniques can be used instead. An alternative technique is the *distance-based hashing* (DBH) proposed by Athisos et al. [10] for efficiently solving the approximate nearest neighbor problem in spaces with arbitrary distance measures. DBH projects the d-dimensional space $\mathbb{X}^d$ into a real line $\mathbb{R}$ using some distance-based hash functions that are defined independently of the similarity measure. DBH is similar to LSH in many aspects but the hash functions for DBH are not necessarily locality-sensitive. Other hashing techniques for efficient similarity search include geometric hashing [12, 22, 30], semantic hashing [47, 48], spectral hashing [62] and similarity hashing [25].

Chapter 3

Region-based Object Discovery

This chapter describes our method for discovering objects from segmented images. Here, each region of the image is regarded as a single object component. We assume that objects are isolated from the background and they do not overlap each other. For the implementation of this method, three kinds of similarity judgments are performed:

- Standard distance-based similarity judgment.
- Distance-based similarity judgment considering the relative size.
- Matching with robustness to small variations.

We propose a modification of the Euclidean LSH scheme described in Sect. 2.2.1 to realize the first two kinds of similarity judgments whereas for the third kind, we extend the standard hashing.

3.1 Proposed Method

In this section, we introduce our method for discovering objects automatically from a set of segmented images Σ. Each image in Σ is represented by a set of regions. The underlying idea is to search for frequent patterns of closely located regions in Σ and consider a frequent pattern as a meaningful object class. Thus, the method runs in four phases described below.

Phase I: By extracting every region in Σ, a set of object components is derived. This set is denoted by $C = \{C_1, C_2, \ldots, C_N\}$.

Phase II: The components in C are classified according to their attributes such as color and size. A label ID is assigned to each component according to the classification result; the labels are expressed by $\ell_1, \ell_2, \ldots, \ell_M$, where M is the number of component classes.

Phase III: Closely located components are gathered to generate object candidates. Let $T = \{T_1, T_2, \ldots, T_Z\}$ be the set of all object candidates.

Phase IV: Object classes are determined by searching frequent patterns in T. A pattern with multiple occurrences is regarded as a meaningful object class. Each object class is represented by the set of component labels that are common among the object candidates of the same class.

Figure 3.1 presents an example of the operation of the object discovery method. This example consists of two images each of which contains an instance of a tree and a house. In total, four objects are contained between the two images: two instances of a tree and two instances of a house. Note that the instances of the house differ in one component (the chimney is present only in the house of the upper image). In this example, 11 components (from C_1 to C_{11}) are extracted from the two images. Next, the labels from ℓ_1 to ℓ_6 are assigned to each component; here, similar components are assigned the same label (e.g. both roofs C_3 and C_8 are assigned the label ℓ_3). Then, the object candidates T_1, T_2, T_3 and T_4 are generated by gathering closely located components. Finally, the Class 1 ("tree") and Class 2 ("house") are regarded as meaningful object classes because both of them have two occurrences in the set of images. The class "tree" is represented by the labels ℓ_1 and ℓ_2 whereas the class "house" is represented by the labels ℓ_3, ℓ_4 and ℓ_5.

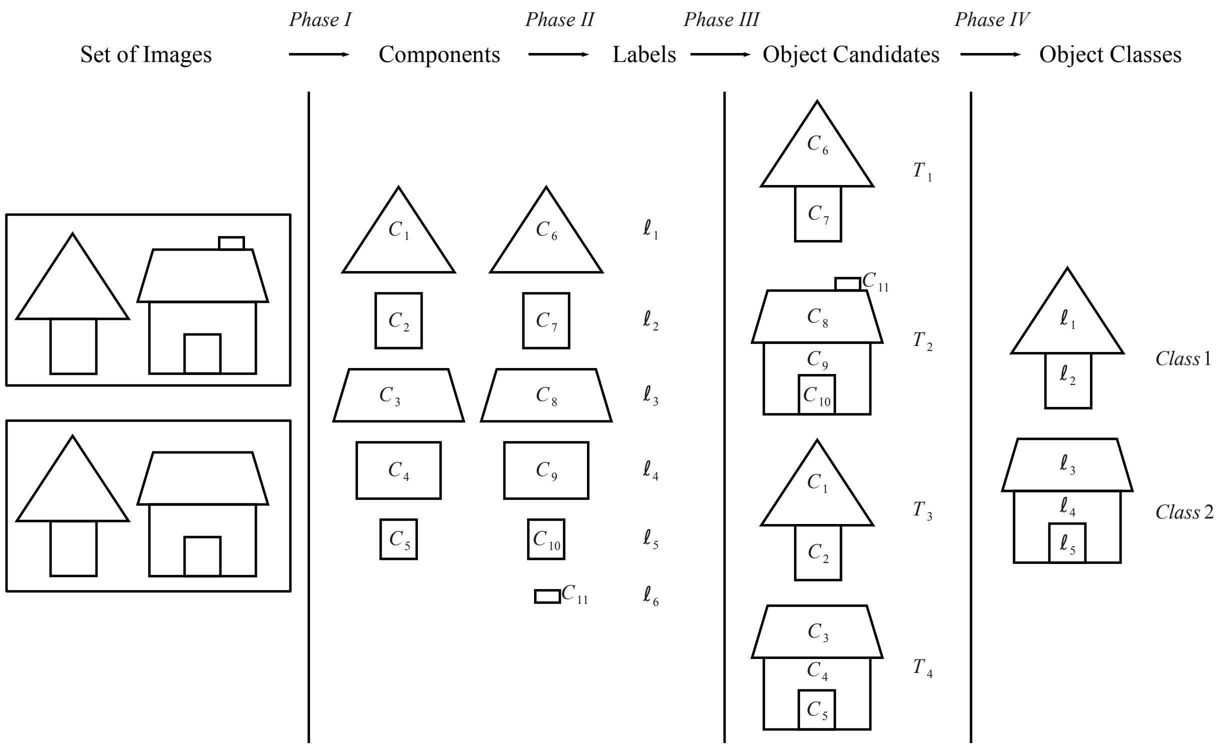


Figure 3.1: Intuitive example of the proposed method.

3.1.1 Phase I: Extraction of Components

Each image in Σ is represented by regions extracted by color segmentation, where each region consists of a group of nearby pixels of the same color. In order to extract object components, regions in Σ that correspond to the background are first removed. Albeit the discrimination between background and foreground regions represents a difficult problem and sometimes requires supervision, in several scenes it is possible to identify the background simply as the extremely big (or poorly textured) regions. All regions in Σ that are not identified as background are regarded as object components.

Since images are represented as multiple regions consisting of pixels with the same color, components can be extracted by clustering near pixels of each color separately. The nearness between pixels is determined by the standard distance-based similarity judgment (Euclidean distance). Here, the Euclidean LSH scheme described in Sect. 2.2.1 is applied to the (X, Y) coordinates of every pixel of each color. To avoid probabilistic fluctuations, we define each tuple $g_1, g_2, \ldots, g_l$ by selecting k bits at equal intervals of a parameter I and prepare I $(l = I)$

different tuples. Consequently, the total number k of selected bits is determined by

$$k = \frac{X_{max} + Y_{max}}{I}, \tag{3.1}$$

where X_{max} and Y_{max} denote respectively the number of columns and rows of the given image. The tuples $g_1, g_2, \ldots, g_I$ are defined so that the k selected bits do not coincide one another at all in the following way:

$$\begin{array}{lcccc} & \multicolumn{4}{c}{\text{Location of selected bits}} \\ g_1 : & I+1, & 2I+1, & \cdots & kI+1 \\ g_2 : & I+2, & 2I+2, & \cdots & kI+2 \\ & \vdots & & & \\ g_I : & 2I, & 3I, & \cdots & (k+1)I \end{array} . \tag{3.2}$$

Note that, since the locations of the hyperplanes are slid by one pixel both in the directions of the x axis and the y axis, the resulting cells are slid in the direction of a vector $(1, 1)$.

In order to group all the pixels of a region into the same cluster, two important properties of Euclidean LSH are taken into account. On one hand, by increasing the number of hyperplanes, pixels stored in the same bucket will become close to each other. On the other hand, by exploiting multiple tuples, near pixels will be stored in the same bucket at least on one hash table. Due to these properties, the clustering algorithm adopts the next rule.

Rule 1 *Pixels stored in the same hash bucket at least on one hash table are classified into the same cluster.*

The above rule is illustrated in Fig. 3.2. Lets assume that we have two tuples g_1 and g_2. Now suppose that the points **A** and **B** enter the same bucket on the hash table of g_1 and the points **A** and **C** enter the same bucket on the hash table for g_2. Then, even if **B** and **C** are distributed to different buckets by g_1 and g_2, **A**, **B** and **C** will form a single cluster altogether.

Now the accuracy of this strategy is discussed from the next two viewpoints, that is, (I) two separate components are not extracted as a single component falsely and (II) a single component

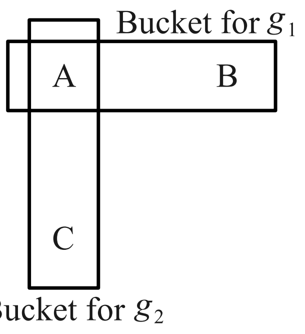


Figure 3.2: Example of the agglomeration step.

is extracted without separating it into pieces.

About (I), as the cells generated by $g_1, g_2, \cdots, g_l$ always become squares with edges of length I, the accuracy of the clustering depends on I. In particular, the next theorem holds.

Theorem 1 *If the minimum distance between two separate components of the same color exceeds $\sqrt{2}I$, these components are not extracted as a single connected component.*

As for (II), for any pixel (X, Y), all the four neighbor pixels $(X + 1, Y)$, $(X - 1, Y)$, $(X, Y + 1)$ and $(X, Y - 1)$ are guaranteed to be stored in the same bucket with (X, Y) for some hash table. Therefore,

Theorem 2 *A single component is never divided into multiple components.*

By applying the above procedure to all images in Σ, a set of components are derived. This set of components is denoted by C.

3.1.2 Phase II: Labeling of Components

The components in C are labeled according to their color and size so that components of the same color with similar size are assigned the same label. Thus, the Euclidean LSH scheme described in Sect. 2.2.1 is applied to the size of the components of the same color. However, since visually large components are less sensitive to variations than small components, the similarity judgment should be relative to the size of the components. Hence, the tuples are defined by selecting k bits at intervals proportional to the distance from the origin. That is, for the i-th tuple $(1 \le i \le l)$, the k bits are selected as follows:

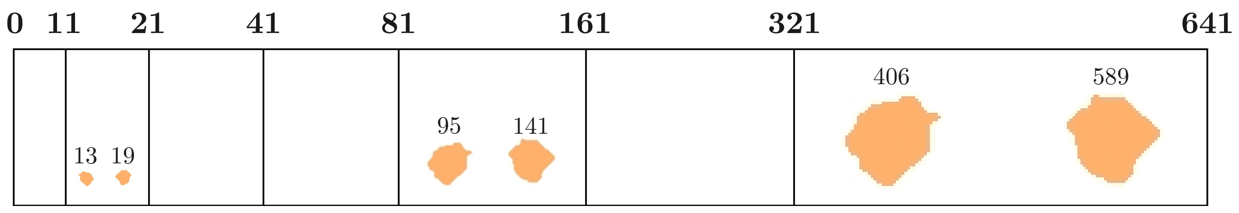


Figure 3.3: Location of selected bits on the real line when $\alpha = 10, \beta = 2$ and $i = 1$.

$$\text{Location of selected bits} \\ g_i : \quad \alpha + i, \quad \alpha\beta + i, \quad \alpha\beta^2 + i, \quad \ldots \quad , \alpha\beta^k + i, \tag{3.3}$$

where α determines the position of the first selected bit and β is the growth factor of the intervals ($\alpha > 0$ and $\beta > 1$). Figure 3.3 illustrates the location of the selected bits on the real line when $\alpha = 10, \beta = 2$ and $i = 1$. Note that the intervals between the selected bits become wider as they become farther from 0. As a result, components whose size difference is small with respect to their sizes will have a high probability of being stored in the same bucket. In Fig. 3.3, the size difference between the two largest components (top right) is large if we compare it with the size of smaller components, however, they lie in the same partition because the difference is small with respect to the size of both components.

To cluster similar components, the CENTER algorithm [27] is applied. CENTER makes graphs where vertices are components and an edge is made between a pair of components if they are stored in the same bucket at least on one hash table. Then, graphs are partitioned in such a way that in each cluster the center node has an edge to the other nodes. This process is carried out by following the next steps.

Step I: For each color, pick up the biggest unchecked component B from the set of components C extracted in Phase I.

Step II: Select all the unchecked components that have an edge to B and merge them into the same cluster.

Step III: Mark all the merged components as checked.

Step IV: Repeat step 1-3 until all the components have been checked.

Figure 3.4 illustrates the operation of the CENTER algorithm. This example shows a graph with 15 nodes. Assuming that $\mathbf{A} \geq \mathbf{B} \geq \mathbf{C} \geq \ldots \geq \mathbf{O}$, three clusters are derived by grouping the nodes adjacent to **A**, **I** and **L** respectively. Note that the node **K** is adjacent to the center **I** and **L** simultaneously but since **I** is bigger than **L**, **K** is assigned to the cluster generated from the center **I** and is not considered for any other cluster.

After this process, the labels $\ell_1, \ell_2, \ldots, \ell_M$ are assigned to the clusters according to the size of the center components such that ℓ_1 and ℓ_M corresponds to the largest and smallest component respectively.

3.1.3 Phase III: Generation of Object Candidates

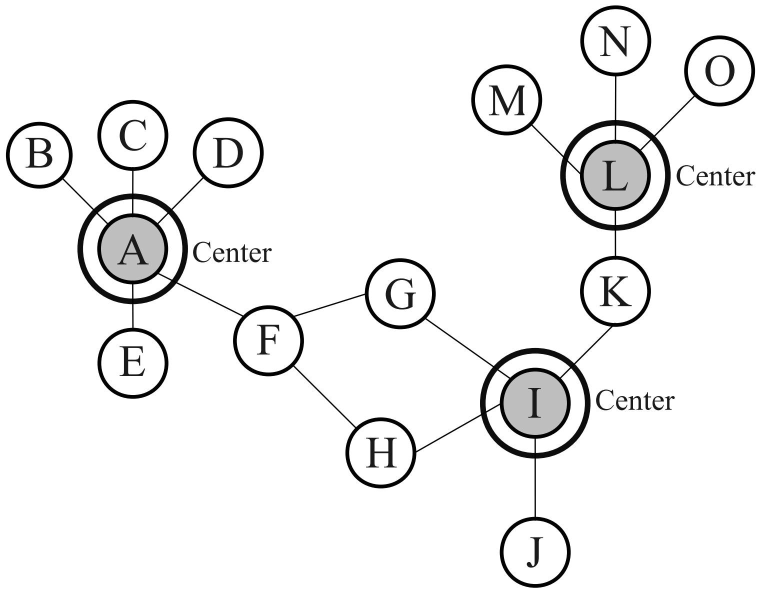
Similar to extracting object components, object candidates are generated by clustering closely located components. Again, the nearness between two separate components is determined by the standard distance-based similarity judgment (Euclidean distance) between their pixels. Therefore, all pixels in every component in C are hashed. As in the extraction of object components, the hash functions are defined by selecting k bits at equal intervals of a parameter I (see Eq. 3.2) and the number k of bits is determined by Eq. 3.1. Thus, to generate object candidates, the next rule is adopted.

Rule 2 *Two separate components C_i and C_j ($i, j = 1, \ldots, N$) are clustered into the same object candidate if one pixel in C_i and one pixel in C_j have the same hash value at least for one hash function.*

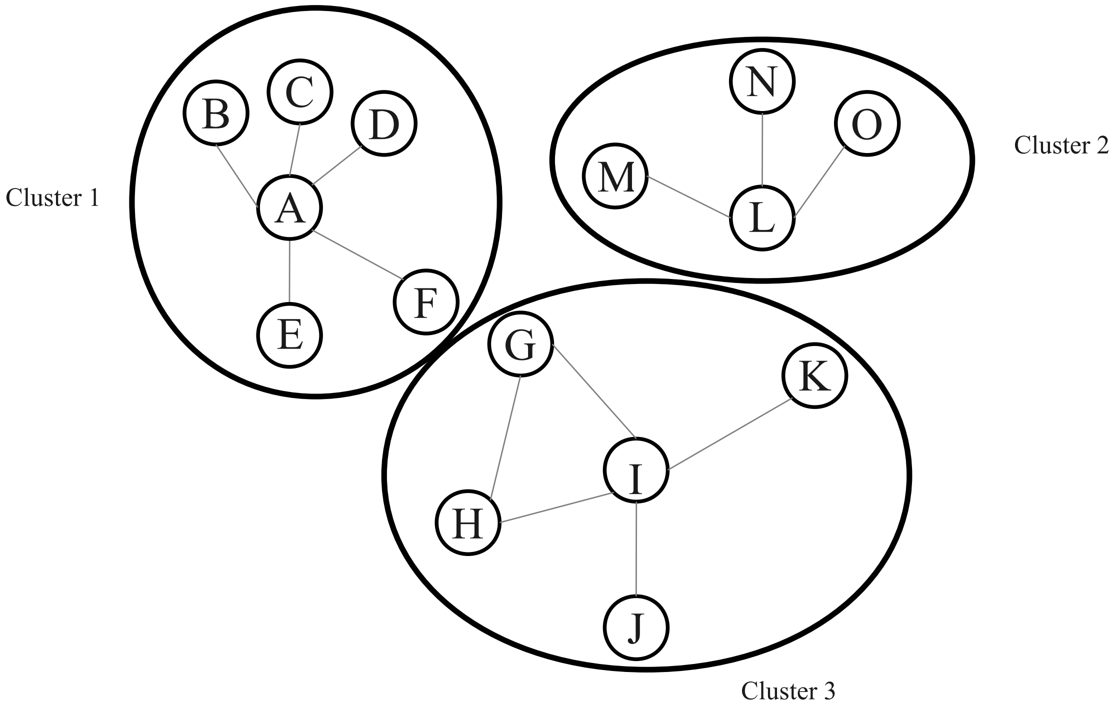
Since object candidates are generated in a similar way that object components are extracted, Theorem 1 can be extrapolated as follows.

Theorem 3 *If the minimum distance between two separate object candidates exceeds $\sqrt{2}I$, these are not merged into a single object candidate.*

Likewise, the parameter I specifies the maximum distance between components to be generated as a single object candidate without dividing it into pieces, that is,



(a)



(b)

Figure 3.4: Operation of the CENTER algorithm [27]: (a) example of a connected component (b) derived clusters.

Theorem 4 *If the minimum distance between a set of components $T \subseteq C$ does not exceed $\sqrt{2}I$, T becomes a single object candidate.*

Each object candidate T_i $(1 \leq i \leq Z)$ is represented by a vector

$$T_i = [\upsilon_1, \dots, \upsilon_M], \tag{3.4}$$

where υ_r $(1 \leq r \leq M)$ denotes the number of components with label ℓ_r in the object candidate T_i. For example, the object candidate T_1 in Fig. 3.1 is generated from the components C_6 and C_7 (with labels ℓ_1 and ℓ_2 respectively) because they are close to each other. In this case, the representation of the object candidate becomes $T_1 = (1, 1, 0, 0, 0, 0)$.

3.1.4 Phase IV: Discovery of Object Classes

In order to discover meaningful object classes, multiple occurrences of similar object candidates are searched. Object candidates are judged as similar if their primary components are the same. The primary components of an object are defined as the large components with respect to the size of object. In Fig. 3.1, the primary components of T_2 are C_8, C_9 and C_{10} (its roof, wall and door respectively) as they are large components with respect to the size of the whole object. In this way, we can match object candidates robustly against small variations as only primary components are taken into account. Standard hashing is applied to accelerate this process. To compute the hash value of an object candidate T_i $(1 \leq i \leq Z)$, the elements υ_r $(1 \leq r \leq M)$ of T_i are first concatenated, that is,

$$cat(T_i) = \upsilon_1 \upsilon_2 \cdots \upsilon_M, \tag{3.5}$$

where $\upsilon_1, \upsilon_2, \cdots, \upsilon_M$ are expressed by λ bits so that $|cat(T_i)| = \lambda M$. In order to avoid small intra-class variations, J hash values are generated for T_i by ignoring the $\xi, \xi + 1, \dots, \xi + J - 1$ smallest components from $cat(T_i)$, where ξ presents the maximum integer such that the sum of the size of the ξ smallest components in T_i does not exceed the $\mu\%$ of the whole size of T_i, that is,

$$\xi \leq \frac{\mu \cdot |T_i|}{100} \tag{3.6}$$

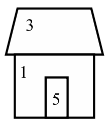
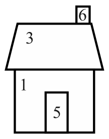
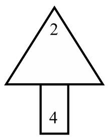
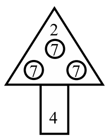
After computing the J hash values for each object candidate, the next rule is used to cluster similar object candidates.

Rule 3 *Two object candidates are classified into the same cluster if at least one of their J hash values is the same.*

After clustering object candidates, only those clusters with multiple object candidates are regarded as meaningful object classes. Each of these classes is represented in the same form as (3.4), where υ_r ($r = 1, \ldots, M$) stands for the number of components with label ℓ_r that are common to all the object candidates of the same cluster. For instance, T_2 and T_4 in Fig. 3.1 are classified into the same cluster by ignoring C_{11}, which is extremely small relative to the size of T_2. Then, since this cluster has two object candidates, it is regarded as a meaningful object class (Class 1) and represented by ℓ_3, ℓ_4 and ℓ_5, i.e., $\upsilon = (0, 0, 1, 1, 1, 0)$. Note that ℓ_6 is not included, because it is not a component of T_4.

Table 3.1 gives an example of the computation of hash values for object candidates. Four object candidates from two object classes (trees and houses) are presented. In this example, the total number of labels is 7 and 2 bits are used to represent each label. Two hash values were generated for each object candidate. In the computation of these hash values, the doors (υ_5) of the two houses, the roof (υ_6) from one house and the apples (υ_7) from one tree were ignored. From the table, we can notice that object candidates of the same class with small variations have at least one identical hash value. Despite slight differences, if we apply Rule 3 to this example, the two houses would be clustered into one class and the two trees would be clustered into another class.

Table 3.1: Generation of hash values from the object candidates.

Object	v_1	v_2	v_3	v_4	v_5	v_6	v_7	Hash Value 1	Hash Value 2
	01	00	01	00	01	00	00	01000100010000	01001000000000
	01	00	01	00	01	01	00	01000100010000	01001000000000
	00	01	00	01	00	00	00	00010001000000	00010001000000
	00	01	00	01	00	00	11	00010001000001	00010001000000

3.2 Experimental Results

For the experiments, each image was segmented by using the MST-based algorithm [58] of Tsunoda et al. and then a color quantization was performed. Finally, the extremely big regions of the image are regarded as background and removed so that objects were isolated. An example of the segmentation, quantization and background removal can be seen in Fig. 3.5.

Initially, we evaluated the robustness of the proposed method against rotation and slide operations. To that end, the proposed method is applied to an image that contains two instances of two different object classes (Fig. 3.5(a)). Note that the orientations of the two instances of the same class differ approximately by 90 degrees. Since the proposed method does not consider the exact location relation between components, both object classes (Fig. 3.6(b) and 3.6(c)) were successfully discovered despite these transformations.

We also evaluated the robustness of our method against intra-class variations. In Fig. 3.7, three examples are presented: human faces and tiger faces (Fig. 3.7(a)), two kinds of candies

(Fig. 3.7(b)) and tea bottles and cans (Fig. 3.7(c)). In all the examples the proposed method derived two object classes successfully. The columns Class 1 and Class 2 in Fig. 3.7 present the instances of each derived class in each image. Note that the two human faces are extracted correctly despite they are from different subjects.

Finally, we applied the method to an image sequence. Figure 3.8 depicts an example of the discovery of a pedestrian from a sequence of three images. This examples suggests that the proposed method can be easily adapted to the discovery of objects from videos.

3.3 Limitations

Although the proposed method achieves a good performance in simple scenes, it can not deal with large amounts of clutter, occlusions and extreme variations of illumination and viewpoint. This limitation stems from two different sources. First, the method assumes that objects are isolated from the background and they do not overlap each other. This assumption must be fulfilled in order to generate meaningful object candidates or otherwise they would be merged. The second limitation is that segmentation results vary greatly under different imaging conditions, which directly affects the performance of the object discovery.

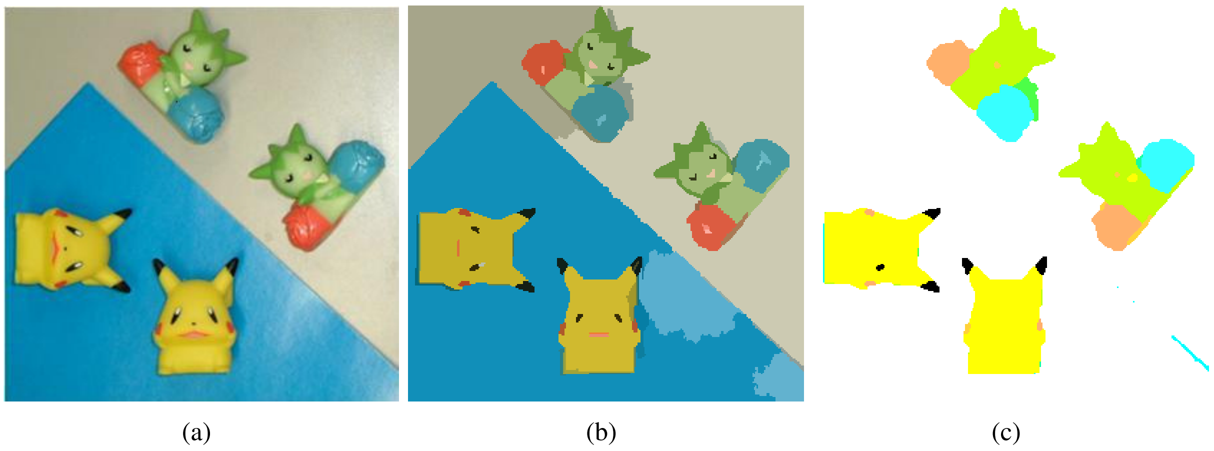


Figure 3.5: Segmentation, quantization and background removal of an image: (a) original image, (b) segmented image, and (c) quantized image with the background removed.

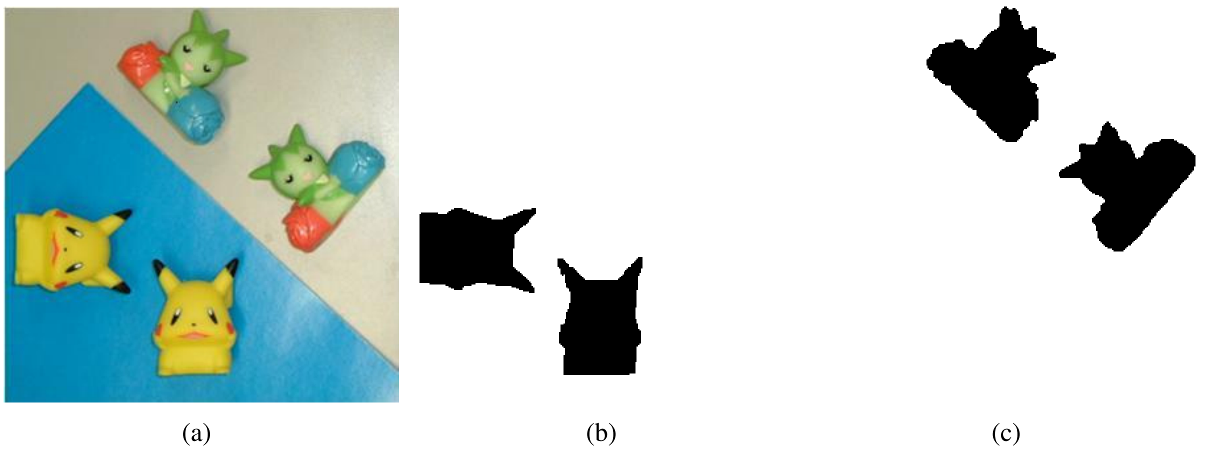


Figure 3.6: Sample results of object discovery against rotation and slide variations: (a) input image, (b) class 1 and (c) class 2.

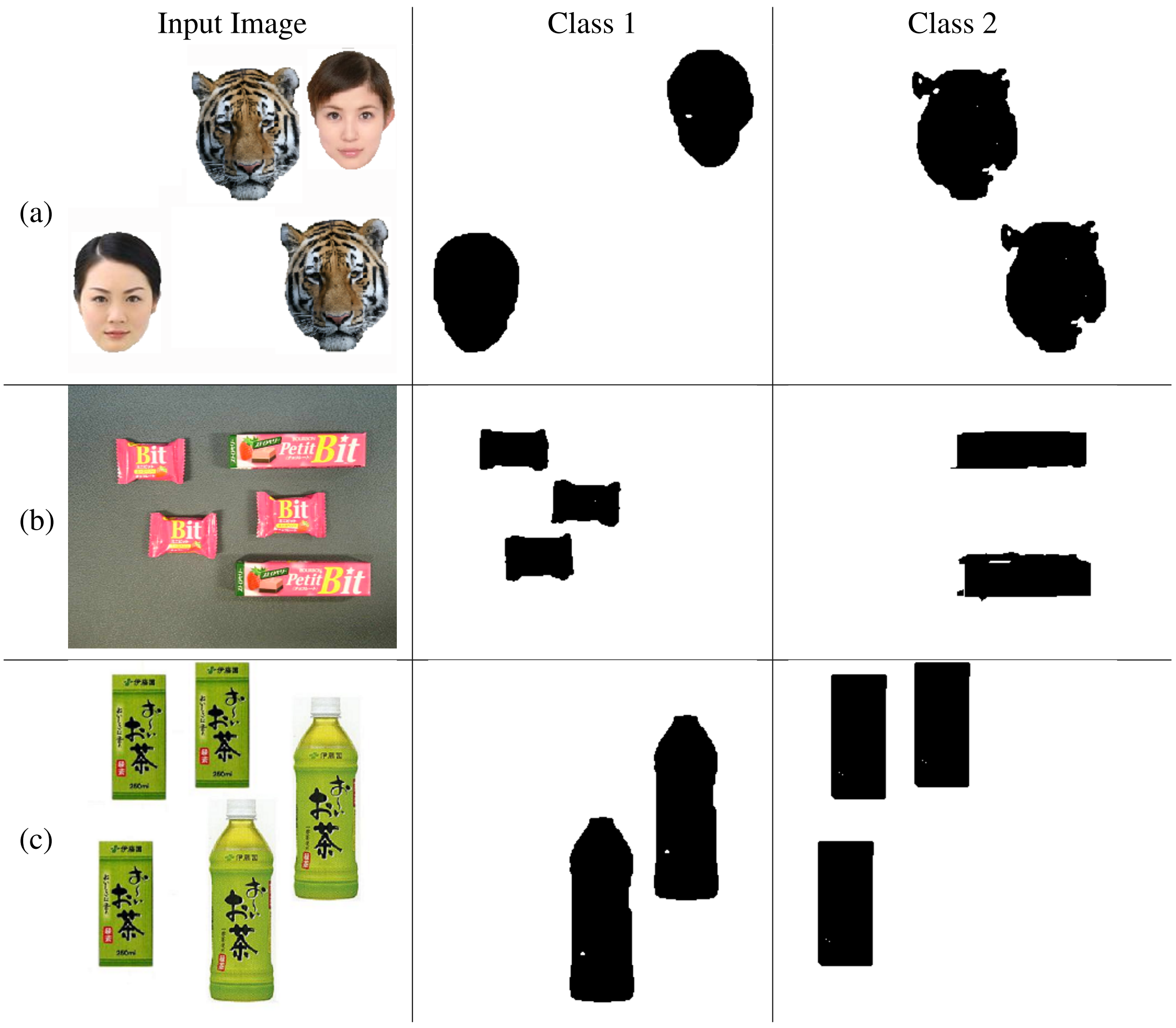


Figure 3.7: Sample results of objects with intra-class variations.

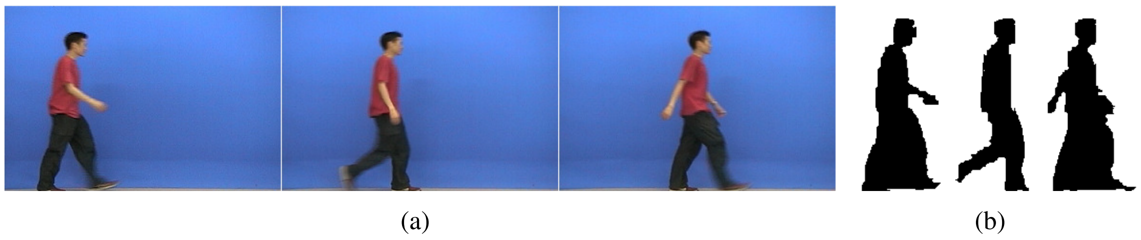


Figure 3.8: Sample result of the discovery of an object from an image sequence: (a) image sequence, (b) discovered object.

Chapter 4

Feature-based Object Discovery

In the previous chapter, we proposed a region-based method that can efficiently discover objects from images without supervision. However, such method is sensitive to clutter, occlusions and variations in imaging conditions, which limits its applicability. In this chapter, we introduce a feature-based method for discovering objects from a given image set. To achieve robustness to variations in imaging conditions, each image is represented by a set of visual words (vector quantized affine covariant features). In addition, to discover objects robustly against occlusions and large amounts of clutter, this method does not generate object candidates by clustering near visual words. Instead, it pays attention to co-occurring visual words. The rationale is that visual words that belong to the same object tend to appear together much more often than those belonging to different objects. Hence, the proposed method yields object models by clustering visual words that consistently appear together in the image set. The generated models are highly discriminative and robust to occlusion, clutter and large variations of scale, illumination and viewpoint. In a quantitative evaluation, this method achieved higher scores than the state-of-the-art. Remarkably, the clustering of such co-occurring visual words is accelerated by Min-Hashing.

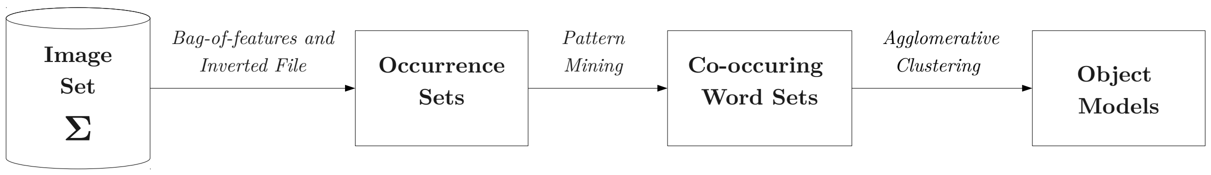


Figure 4.1: Overview of the object discovery process.

4.1 Proposed Method

This section introduces our method for discovering objects from a given image set $\Sigma = \{I_1, I_2, \ldots, I_N\}$. We call this method *Min-Hashing Based Object Discovery* (or MHOD for short). This method consists of the following three phases:

Phase I: Representing each image with a BOF model and indexing the set of BOF models with an inverted file.

Phase II: Mining co-occurring word sets from the inverted file.

Phase III: Generating object models by clustering co-occurring word sets based on the number of common visual words.

The overview of the object discovery process is illustrated in Fig. 4.1. A remarkable characteristic of MHOD is that it does not require the number of clusters (kinds of objects) or other type of supervision but it exploits the co-occurrence between visual words and the similarity between co-occurring word sets to generate the object models automatically. In the following, we discuss in detail each phase of MHOD.

4.1.1 Phase I: Bag-of-Features and Inverted File

MHOD exploits the bag-of-features (BOF) approach to represent each image. The set of BOF models is further indexed with an inverted file. Next, we review the steps to obtain such a representation.

Interest Point Detection

In the BOF approach, local image features are first extracted for each image. Local features, as opposed to global features, can cope with partial occlusions and clutter because objects are represented not only by a single feature but by a set of small features localized in different parts of the object. In this work, we extract affine covariant features which are invariant to affine transformations and robust to viewpoint and illumination changes. In particular, we apply the hessian-affine detector [39] for finding blob-like patches in the images. The hessian-affine detector uses the determinant of the Hessian matrix to first find interest points in the scale-space at location and scales of a local structure for which a given function attains an extremum. At each detected point, an elliptical patch is iteratively adapted so that its size and shape vary covariantly with affine transformations. This is achieved by finding the transformation that projects the affine pattern to one with equal eigenvalues in the second moment matrix of the intensity gradient. Finally, to achieve invariance to rotation, the shape of each elliptical patch is normalized by transforming it into a circle. Figure 4.2 shows the hessian affine patches detected in two different images of the same object. Note that despite variations, several similar patches are detected in both images.

Local Feature Description

Each detected local image patch is encoded based on its appearance by using a patch descriptor. Here, we use the SIFT descriptor proposed by Lowe [38]. This descriptor first divides the patch into a 4×4 grid cell (Fig. 4.3 (a)). Then, it quantizes the gradient orientations into 8 different orientations (Fig‘4.3 (b)) and forms a histogram for each cell. The result is a 128-dimensional vector. To achieve invariance to rotation changes, the orientations and magnitudes are normalized by computing the gradients relative to the dominant orientation of the patch (highest peak among all the gradient orientations within the patch.).

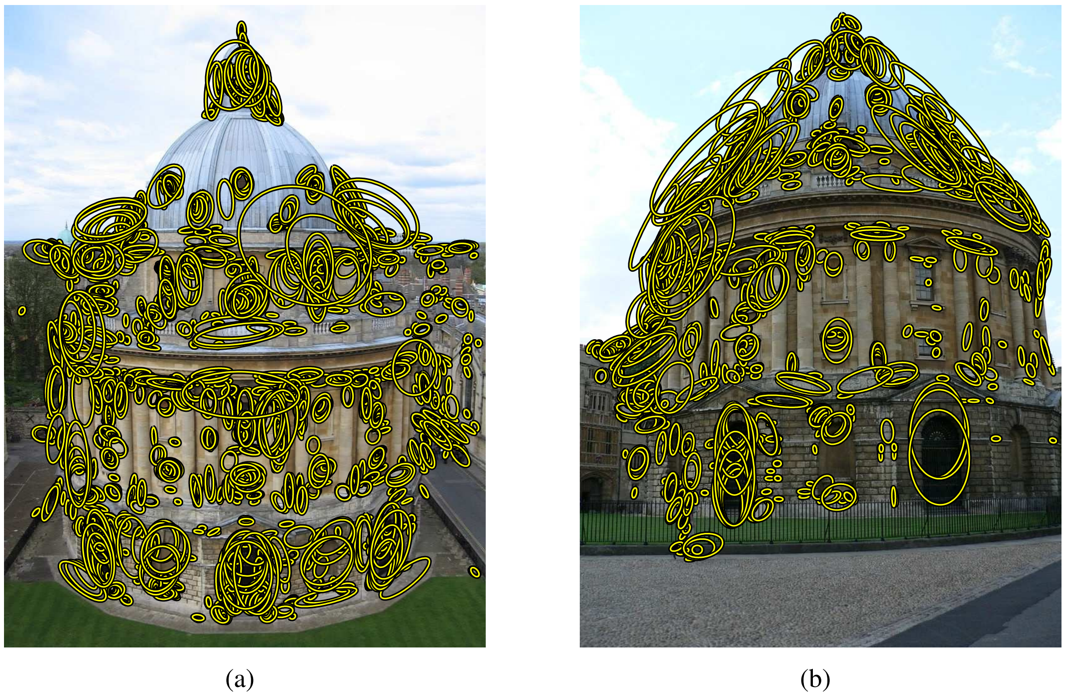


Figure 4.2: Hessian-Affine elliptical patches of two different images of the same object.

Feature Quantization

Now that each image is represented by a set of 128-dimensional vectors, a vocabulary of *visual words* $V = \{v_1, \ldots, v_M\}$ is constructed by clustering the local image features in all the images. The size of the vocabulary size is a trade-off between the discrimination power and the repeatability of the visual words. Originally, k-means was used to cluster the feature vectors. However, for large vocabulary sizes, k-means can be prohibitely expensive. To overcome this problem, some variants that scale well with the vocabulary size have been proposed [40, 42].

Each local image feature is assigned the ID of the nearest visual word. In standard BOF, each image is described as a frequency vector of visual words. However, as the proposed method only needs to analyze the occurrence pattern of the visual words, only the presence or absence of the visual word is recorded. In other words, each image is represented by a set of visual words. This results in a more compact representation with a good discrimination power for large vocabularies. In fact, it has been shown [31] that for vocabularies larger than 10000 visual words, the binary BOF slightly outperforms standard BOF in search quality.

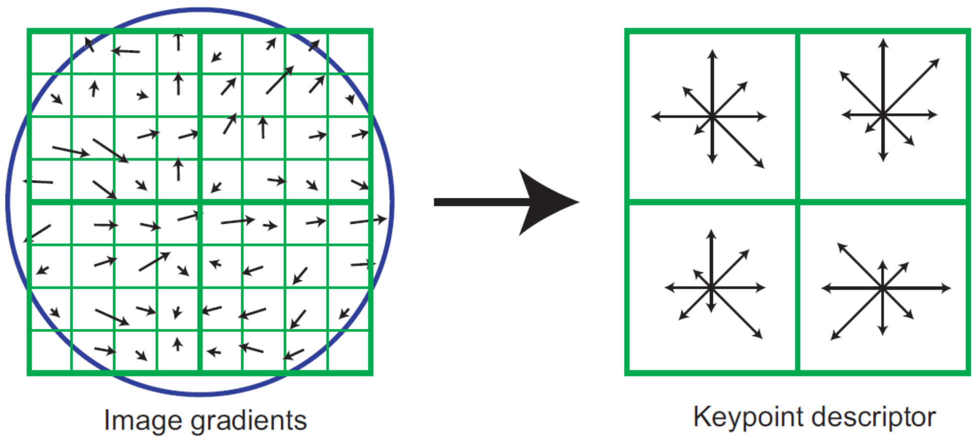


Figure 4.3: SIFT descriptor [38]: gradient magnitudes and orientations of an image patch (the circle indicates the Gaussian center-weighting) divided into a 4×4 grid cell (left). Gradient orientations quantized into 8 orientations for each cell (right).

Stop List

A stop list is used to discard very rare and very common visual words. Very common visual words appear in images of many different objects and therefore are not discriminative. On the other hand, very rare visual words are not relevant for object discovery as they only occur incidentally in Σ. Sivic and Zisserman [51] reported that an adequate stop list can reduce the number mismatches when retrieving similar frames of a video.

Inverted File Indexing

The set of BOF models is indexed with an inverted file structure. For each visual word v_i, the inverted file stores a set whose elements are the IDs of the images in which v_i occurs. We denote the set of images containing v_i by $\hat{v}_i$ and refer to it as the *occurrence set* of v_i. Furthermore, all the occurrence sets on the inverted file are denoted by $\hat{V}$. We use these sets for finding co-occurring visual words.

4.1.2 Phase II: Co-occurring Word Set Mining

Now that each visual word v_i is associated with the occurrence set $\hat{v}_i$, we can compute the similarity between v_i and v_j by applying Min-Hashing to $\hat{v}_i$ and $\hat{v}_j$. Since $\hat{v}_i$ presents the set of images in which v_i occurs, the Jaccard similarity $sim(\hat{v}_i, \hat{v}_j)$ measures how often v_i and v_j

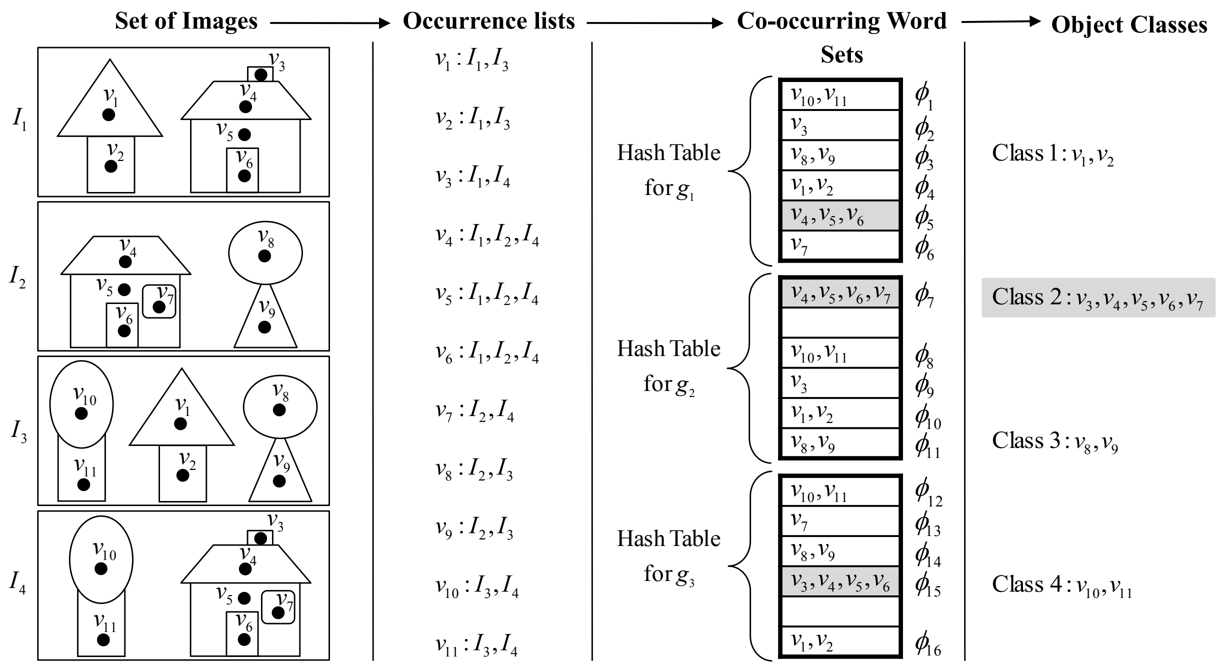


Figure 4.4: Toy-example of the object discovery process by MHOD.

co-occur in the image set Σ. So, for a given visual word v_i , we exploit Min-Hashing to search other visual words which tend to co-occur often with v_i in Σ.

The min-hash value of a visual word v_i is defined as

$$h(v_i) = min(\pi(\hat{v}_i)). \tag{4.1}$$

As mentioned in Sect. 2.2.1, we rely on multiple min-hash functions chosen independently at random. That is, we define l tuples g_i $(1 \leq i \leq l)$ each of which consists of r min-hash values. A set of visual words that enter the same bucket on one of the hash tables is called a *co-occurring word set* and denoted by ϕ. Here, one co-occurring word set ϕ is derived from one bucket storing multiple visual words. We expect that discriminative visual words that belong to the same object enter the same bucket and form a co-occurring word set ϕ, as they should appear together in the images that contain the object. By contrast, unrelated visual words from different objects will not be stored in the same bucket.

Given a set of d visuals words $\{v^1, v^2, \ldots v^d\}$, the probability that all the d visual words take the same min-hash value for a single min-hash function h, i.e, $P[h(v^1) = h(v^2) = \cdots = h(v^d)]$ is given by Eq. (4.2).

$$P[h(v^1) = h(v^2) = \cdots = h(v^d)] = \frac{|\hat{v^1} \cap \hat{v^2} \cap \cdots \cap \hat{v^d}|}{|\hat{v^1} \cup \hat{v^2} \cup \cdots \cup \hat{v^d}|}. \quad (4.2)$$

In Eq. (4.2), the numerator presents the number of images which contain all the d visual words, whereas the denominator corresponds to the number of images which include at least one of the d visual words. As the visual words appear in the same images more frequently, the value of Eq. (4.2) increases, since its numerator becomes larger. This implies that the d visual words are more likely to become a co-occurring word set, as their occurrence pattern in the image set Σ grows more positively correlated.

Pruning

Due to the random nature of Min-Hashing, some co-occurring word sets can contain noisy (unrelated) visual words. To get rid of such visual words, we perform the following pruning step. Given a co-occurring word set denoted by ϕ, we first scan the inverted file to obtain a list of images $Q(\phi)$ that contains at least $\alpha|\phi|$ visual words in ϕ ($0 < \alpha \leq 1$). Then, the visual words that occur in less than $\beta|Q(\phi)|$ images of $Q(\phi)$ ($0 < \beta \leq 1$) are discarded from ϕ. Finally, we remove ϕ completely if it contains very few visual words after discarding visual words. We also remove ϕ if $|Q(\phi)|$ is small as it may contain visual words that originate from different objects and appear together incidentally.

4.1.3 Phase III: Agglomerative Clustering

Because of unstable and polysemous visual words, the co-occurring word sets extracted by Min-Hashing will represent only a part of the entire object model. By contrast, highly stable visual words will be contained together in multiple co-occurring word sets.

Let us illustrate this phenomenon with the toy-example in Fig. 4.4. This example consists of 4 images and a visual vocabulary of 11 visual words. Each visual word v_i is registered to the 3 hash tables corresponding to the tuples g_1, g_2 and g_3, computed from each occurrence set $\hat{v}_i$.

Then, 16 co-occurring word sets from ϕ_1 to ϕ_{16} are extracted from the hash tables. As we can observe, stable visual words belonging to the same object are mapped to the same co-occurring word set often. For example, consider the object "house" composed of the visual words v_3, v_4, v_5, v_6 and v_7. As v_4, v_5 and v_6 always appear together, they are included in the same co-occurring word set three times (ϕ_5, ϕ_7 and ϕ_{15}). On the other hand, unstable visual words are mapped to different co-occurring word sets, even if they belong to the same object. In Fig. 4.4, v_3 and v_7 are never contained in the same co-occurring word set because they appear together only once in I_4. We can also observe that ϕ_5, ϕ_7 and ϕ_{15} share the stable visual words v_4, v_5 and v_6 and contain other informative visual words (v_3 and v_7).

Motivated by the above observations, so as to obtain more representative object models, we merge co-occurring word sets that share many visual words in an agglomerative manner. Because of the agglomerative clustering, the number of object kinds need not be specified in MHOD. Let ϕ_i and ϕ_j be two co-occurring word sets. Note that the elements of the two sets are visual words. We measure the proportion of visual words shared between ϕ_i and ϕ_j by their overlap coefficient in Eq. 4.3.

$$ovr(\phi_i, \phi_j) = \frac{\mid \phi_i \cap \phi_j \mid}{min(\mid \phi_i \mid, \mid \phi_j \mid)} \in [0, 1]. \tag{4.3}$$

Then, if $ovr(\phi_i, \phi_j) > \epsilon$, we merge ϕ_i and ϕ_j into the same cluster, where ϵ is a parameter of the algorithm. We can rely on Min-Hashing to find the co-occurring word sets to be merged promptly. Since

$$ovr(\phi_i, \phi_j) = \frac{\mid \phi_i \cap \phi_j \mid}{min(\mid \phi_i \mid, \mid \phi_j \mid)} \geq \frac{\mid \phi_i \cap \phi_j \mid}{\mid \phi_i \cup \phi_j \mid} = sim(\phi_i, \phi_j),$$

a pair of co-occurring word sets whose Jaccard similarity is high will also have a large overlap coefficient. Hence, we may judge whether a pair of co-occurring word sets potentially take a high overlap coefficient from the fact that they enter the same hash bucket in Min-Hashing. This strategy avoids the overhead of computing the overlap coefficient between all the pairs of co-occurring word sets. We remark here that Min-Hashing is applied to a set of visual words in this step, whereas it is applied to a set of images in the co-occurring word set mining of Sect. 4.1.2. The min-hash value for ϕ_i becomes its first visual word after the order of all the visual words is permuted by the permutation π randomly chosen. That is,

$$h(\phi_i) = min(\pi(\phi_i)). \tag{4.4}$$

Again, we use multiple min-hash values to construct l hash tables. Two co-occurring word sets that share many visual words are expected to enter the same bucket at least on one hash table.

Our algorithm to cluster co-occurring word sets agglomeratively consists of the following 5 steps.

1. Each co-occurring word set is stored into l hash tables.

2. If two co-occurring word sets ϕ_i, ϕ_j are stored in the same bucket on some hash table, they are regarded as a candidate pair to be merged.

3. For every candidate pair of co-occurring word sets (ϕ_i, ϕ_j), we compute their overlap coefficient as
$$ovr(\phi_i, \phi_j) = \frac{\mid \phi_i \cap \phi_j \mid}{min(\mid \phi_i \mid, \mid \phi_j \mid)} \in [0, 1].$$

4. We construct a graph G such that each co-occurring word set ϕ_i becomes a node and an edge is constructed between each candidate pair of co-occurring word sets ϕ_i, ϕ_j with $ovr(\phi_i, \phi_j) > \epsilon$.

5. We compute all the connected components in G. Co-occurring word sets (i.e. vertices) belonging to the same connected component are merged into a single cluster and all their visual words become the final object model.

With this algorithm, chains of co-occurring word set pairs with high overlap coefficient are merged into the same cluster. As a result, co-occurring word sets associated with the same object will belong to the same cluster even if they share very few or no visual words, so long as they are members of the chain. For example, consider three co-occurring word sets ϕ_i, ϕ_j and ϕ_k associated with the same object. Even if ϕ_i and ϕ_j do not share visual words at all, they will be merged into the same cluster, in case ϕ_k shares many visual words with both ϕ_i and ϕ_j. In general, the generated clusters have the property that for any co-occurring word set in a cluster, there exists at least one co-occurring word set in the same cluster with which it has an overlap coefficient greater than ϵ. Conversely, two co-occurring word sets have an overlap coefficient less than ϵ, if they belong to different clusters.

In the example in Fig 4.1, the agglomerative clustering on the co-occurring word sets produces 4 object models (from Model 1 to Model 4). Here, ϕ_5, ϕ_7 and ϕ_{15} are merged into the same cluster to form Model 3, because they share the stable visual words v_4, v_5 and v_6. In this case, the object model consists of the visual words contained in either ϕ_5, ϕ_7 or ϕ_{15}, i.e., v_3, v_4, v_5, v_6 and v_7. Despite v_3 and v_7 are never contained in the same co-occurring word set, they are correctly assigned to the same object model by the agglomerative clustering.

4.2 Retrieval

After performing the agglomerative clustering, we obtain a set of object models which are composed of the visual words contained in the co-occurring word sets of the same cluster. Since images are also represented as sets of visual words, we can determine whether an image contains a specific object from the number of visual words shared between the object model and the image. Especially, we can efficiently identify all the images that share visual words with the object model by searching the occurrence sets of the visual words in the object model. Next, by investigating the number of visual words shared with these images, we retrieve images that share many visual words with the object model and therefore are likely to contain the object. The retrieved images can be further ranked according to the number of shared visual words in order to show the most relevant images first.

4.3 Scalability

In order to achieve scalability with regard to execution time, we generate object models by simply analyzing the occurrence pattern of visual words. In fact, MHOD only searches for similar occurrence sets on the inverted file. This contrasts to other methods that adopt expensive learning algorithms. In addition, the most time-consuming tasks of MHOD, namely mining and clustering co-occurring word sets, are efficiently performed by Min-Hashing, which has proved to be particularly suitable for handling large datasets (see [16, 17, 27]). The time to compute a min-hash value for a set is linear to the number of elements in the set, since we need to find the minimum from the numbers assigned to all its elements. Now, consider the time complexity for the co-occurring word set mining which is the most time consuming part of MHOD. Here, the time to compute $r \cdot l$ min-hash values for $|V|$ visual words becomes $O(r \cdot l \cdot W \cdot |V|)$, where W is the average number of images in the occurrence sets. In addition, before the computation of min-hash values, a time of $O(r \cdot l \cdot |\Sigma|)$ is incurred to generate $r \cdot l$ randomly chosen permutations of the image set Σ. Therefore, the total time complexity for the co-occurring word set mining grows to $O(r \cdot l \cdot (W \cdot |V| + |\Sigma|))$. Because $W \ll |\Sigma|$ in general, this time complexity is linear to the number of images, which shows the scalability of MHOD. A similar analysis can be done for the agglomerative clustering. On the other hand, as object models are represented as sets of visual words, we can also retrieve the images that contain a particular object quite fast by searching the occurrence of the object model in the inverted file as explained in Sect. 4.2.

As for memory consumption, it has been pointed out that Min-Hashing consumes much memory to store all the hash tables. However, both for mining and clustering co-occurring word sets, we only need to store one hash table at a time. Hence, we can avoid the high space complexity often associated with Min-Hashing.

Thus, the proposed method can be applied to both huge image sets and large visual vocabularies.

4.4 Experiments

In this section, MHOD is demonstrated on the Oxford buildings dataset [4]. First, we evaluate the results qualitatively by visually examining the discovered objects. In particular, we analyze the meaningfulness and discrimination power of the generated object models. We also carry out a quantitative evaluation using a set of ground truth landmarks and compare our results with the state-of-the-art methods. Finally, we analyze the time and space efficiency of MHOD.

4.4.1 Setup

Oxford Buildings Dataset

This dataset consists of 5,062 images retrieved from Flickr [5] using particular Oxford landmarks as queries (e.g. "All Souls Oxford"). Random image samples from the Oxford buildings dataset are shown in Fig. 4.5. Due to inaccurate annotations, several unrelated images (which serve as distractors) are also contained in the dataset. For each image, affine covariant hessian patches [39] are detected. Each detected patch is represented by a SIFT vector [38]. The total number of detected patches over all the images is 16,334,970. These 16 million SIFT vectors are classified into 1 million visual words by the approximate k-means clustering of Philbin et al. [42]. The reason why we set the size of visual vocabulary to 1 million is that [42] reported that this value yields the best performance. We used the files with precomputed visual word IDs and geometries available at [4] for constructing the BOF models and the inverted file; visual words that occurred in more than 30% or less than 0.1% of the images in the dataset were discarded by the stop list.

Manually generated annotations for the occurrence of 11 Oxford landmarks (see Fig. 4.6) are also provided as ground truth at [4]. In addition, images with the same landmark annotation are assigned one of the following three labels.

- *Good:* a nice, clear picture of the object/building.
- *OK:* more than 25% of the object is clearly visible.

Table 4.1: Number of annotated images for each ground truth landmark.

Ground truth	Good	OK	Junk
All Souls	24	54	33
Ashmolean	12	13	6
Balliol	5	7	6
Bodleian	13	11	6
Christ Church	51	27	55
Cornmarket	5	4	4
Hertford	35	19	7
Keble	6	1	4
Magdalen	13	41	49
Pitt Rivers	3	3	2
Radcliffe Camera	105	116	127
Total	272	296	299

- *Junk:* less than 25% of the object is visible, or there is a very high level of occlusion or distortion.

In Table 4.1, we present the number of images with each label for each ground truth landmark. Note that some ground truth landmarks have much more images than others.

Parameter tunings

The parameters of MHOD were set as follows. To mine co-occurring word sets, we used 532 tuples, each of which was composed of 4 min-hash values. With these parameter values, the turning point of the unit step function becomes $s* \approx 0.19$. Pruning co-occurring word sets was realized with the parameters $\alpha = 0.7$ and $\beta = 0.8$; co-occurring word sets with less than 3 visual words or appearing in fewer than 3 images were removed. For the agglomerative clustering, we used 255 tuples of 3 min-hash values each ($s* \approx 0.13$) and the threshold ϵ for the overlap coefficient was set to 0.6.

Rankings

We define two kinds of rankings to examine and evaluate the results: one over the images that contain the discovered object and another over the discovered objects themselves. For the

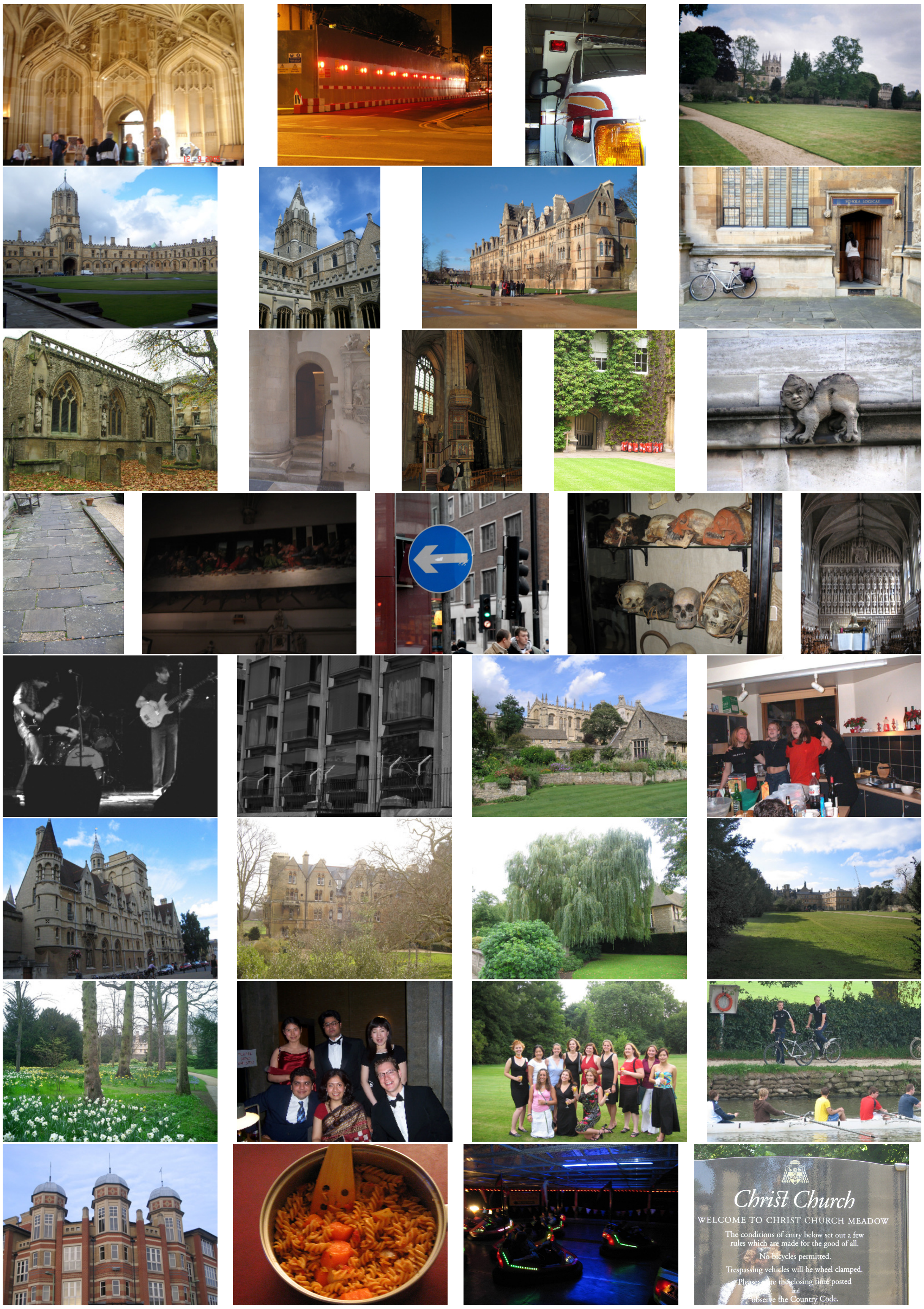

Christ Church
WELCOME TO CHRIST CHURCH MEADOW
The conditions of entry below set out a few rules which are made for the good of all.
No bicycles permitted.
Trespassing vehicles will be wheel clamped.
observe the Country Code.

Figure 4.5: Random image samples from the Oxford buildings dataset.

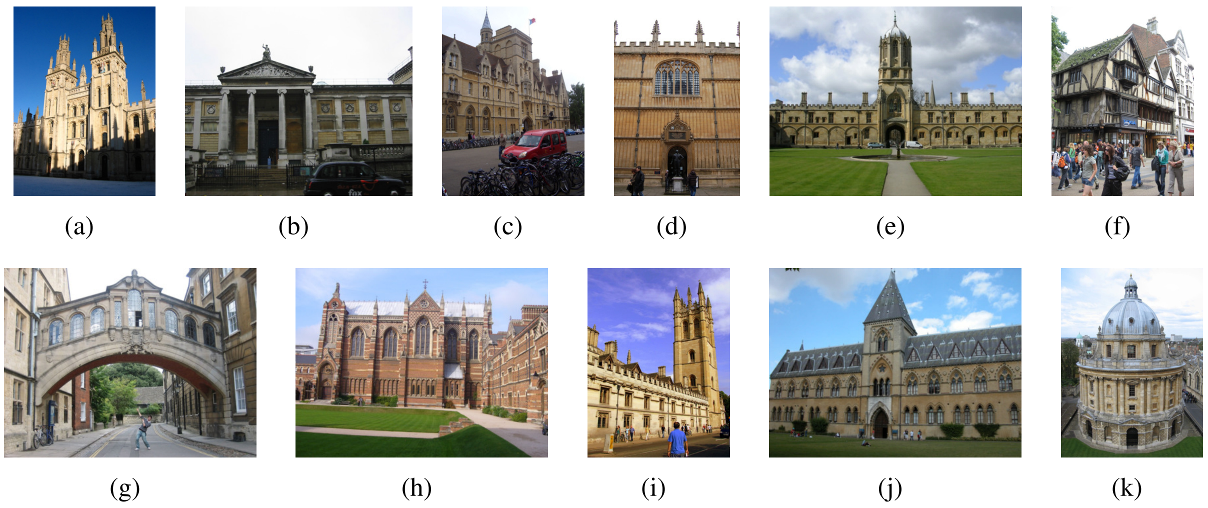


Figure 4.6: Ground truth landmarks of the Oxford buildings dataset: (a) All souls, (b) Ashmolean, (c) Balliol, (d) Bodleian, (e) Christ Church, (f) Cornmarket, (g) Hertford, (h) Keble, (i) Magdalen, (j) Pitt Rivers and (k) Radcliffe Camera.

image ranking, we use each object model (set of visual words) to query the image set through the inverted file. Each query yields a list of images that contains a particular object. Then, the images in the list are ranked based on the number of shared visual words so that more relevant images have higher rank. For the object ranking, we rank the discovered objects according to the size of their models (number of visual words) so that more representative objects have higher rank. Fig. 4.7 illustrates the top-10 objects in the object ranking. Interestingly, the top-5 objects correspond to ground truth landmarks (compare Fig. 4.6 and Fig. 4.7).

Methodology

In our experiments, we apply MHOD to all the 5,062 images of the Oxford buildings database to extract object models. Then, we use each object model to retrieve the images with the corresponding object. We further rank the retrieved images according to the ranking in Sect. 4.4.1. We will confirm the meaningfulness and robustness of derived object models from the fact that the objects annotated by humans are retrieved with high accuracy using the object models. Here, the accuracy is evaluated by the ranking result.

We do not split the dataset in a training and a test set. However, this is also the case for other state-of-the-art methods such as [43] and [44]. Because our primary goal is to extract

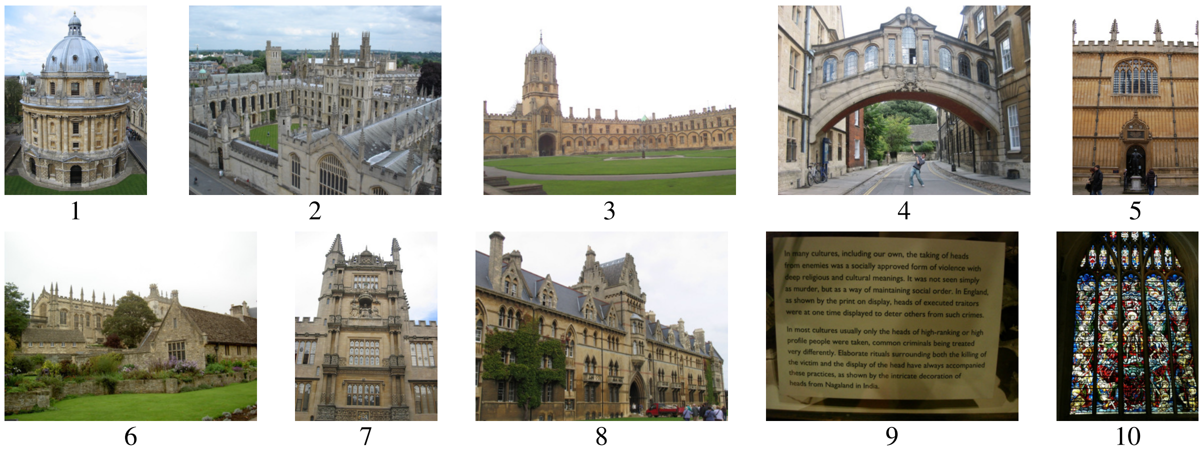


Figure 4.7: Top-10 ranked objects. The rank of the object is shown below each image.

meaningful object models from a set of images automatically, our experiments focus on the meaningfulness of the derived object models rather than on the ability to recognize unseen new views of an object.

4.4.2 Results

Qualitative Evaluation

Several different objects were discovered by MHOD, including objects corresponding to the 11 ground truth landmarks[1]. Figure 4.8 shows typical image samples of the top ranked objects associated with All Souls, Christ Church, Hertford and Radcliffe Camera. The samples displayed in Fig. 4.8 are presented in descending order of rank: from the top-ranked images (left) to lower-ranked images (right). Although not presented here, all the high-ranked images are similar to the examples shown here. Note that the matched affine covariant features within each image are correctly localized on the corresponding object (even in the lower-ranked images) despite occlusions, clutter and extreme variations of scale, illumination and viewpoint. These examples demonstrate the meaningfulness and robustness of the object models. A quantitative evaluation using the ground truth landmarks is given in Sect. 4.4.2.

As mentioned before, in MHOD, the number of different kinds of objects is not fixed but rather depends on the correlation of the visual word occurrences and the similarity of co-

[1]Some ground truth landmarks had more than one associated object.

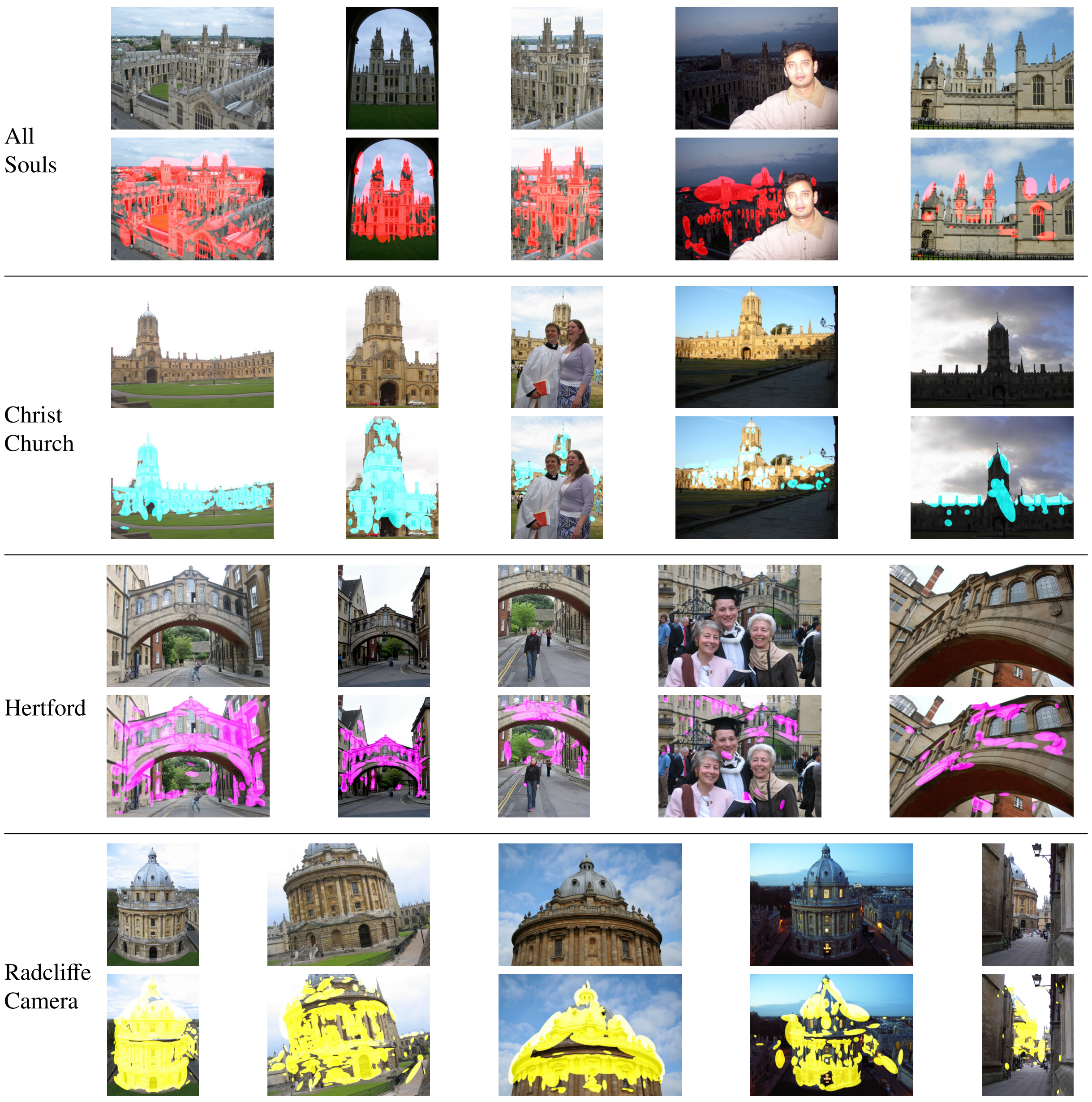


Figure 4.8: Sample images of four objects corresponding to ground truth landmarks. The original images (top) and the images with the matched affine covariant regions displayed (bottom) are presented for each object.

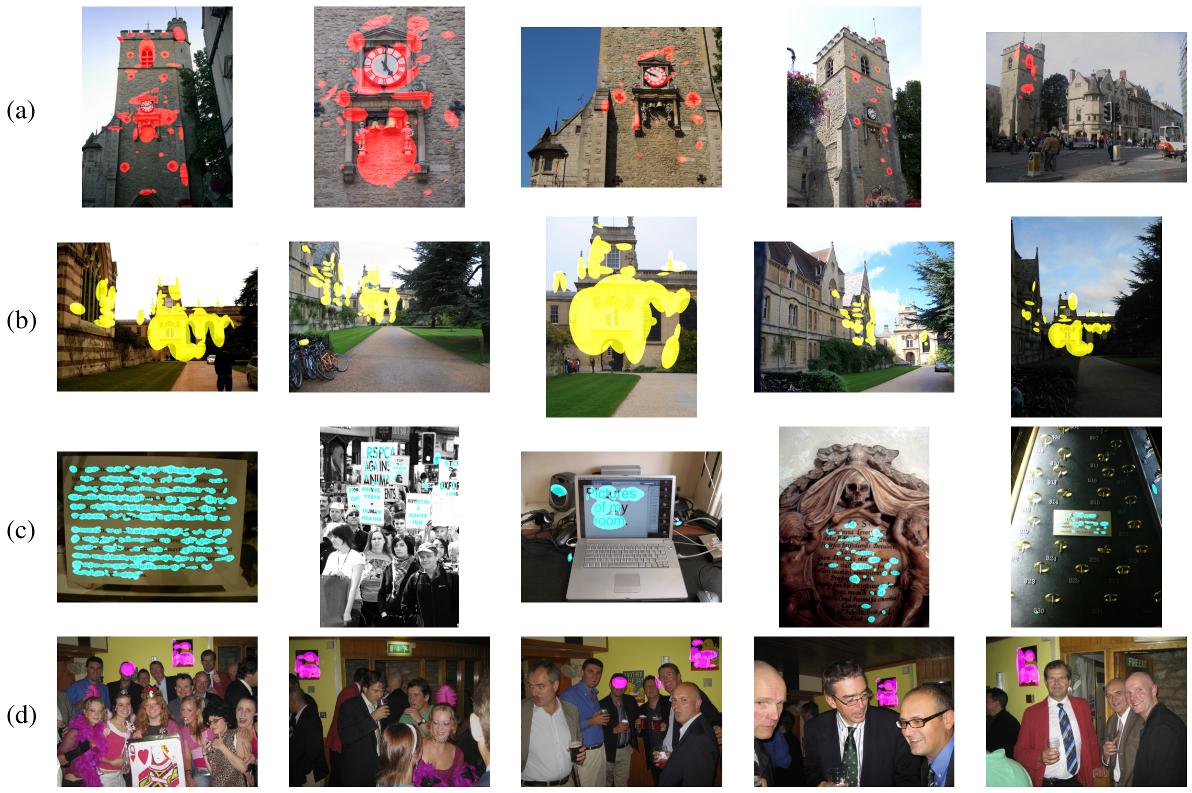


Figure 4.9: Sample images of four objects not associated with ground truth landmarks: (a) St Michael at the North Gate, (b) Trinity College, (c) black letters over light background and (c) a cartoon picture on a wall.

occurring word sets. As a consequence, many objects different from the ground truth were also discovered. Four examples of such objects are illustrated in Fig. 4.9. The rows (a) and (b) correspond to other Oxford landmarks whereas (c) and (d) rows are non-building objects, namely dark letters over light background and a cartoon picture on a wall. Notice that the cartoon picture is quite small relatively to the image size. This shows that MHOD can discover objects even if they cover only a small portion of the images.

Remarkably, different objects that appear in some images together were correctly discriminated (see Fig. 4.10). Again, the matched affine covariant features are mostly localized on the corresponding object which shows that MHOD generates highly discriminative models.

Quantitative Evaluation

To evaluate the performance of MHOD quantitatively, we compute the precision-recall curve from the ranked image list of the object model with the highest rank for each ground truth land-

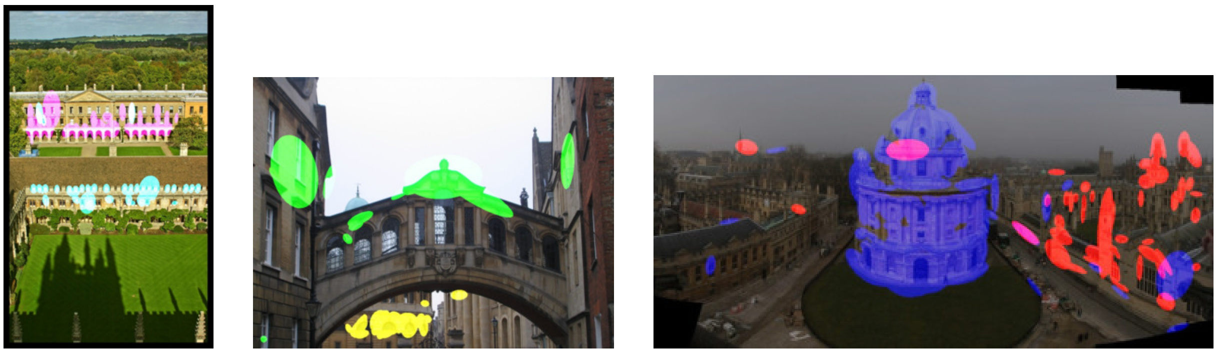

(a) (b) (c)

Figure 4.10: Sample images containing multiple discovered objects (a) All Souls and Radcliffe Camera, (b) Magdalen Cloisters and New Building and (c) Hertford Bridge and Sheldonian Theater.

mark, where precision is the ratio of retrieved positive images to the total number of retrieved images and recall is the ratio of retrieved positive images to the total number of positive images. Namely,

$$\text{precision} = \frac{|\{\text{positive retrived images}\}|}{|\{\text{retrieved images}\}|} \tag{4.5}$$

and

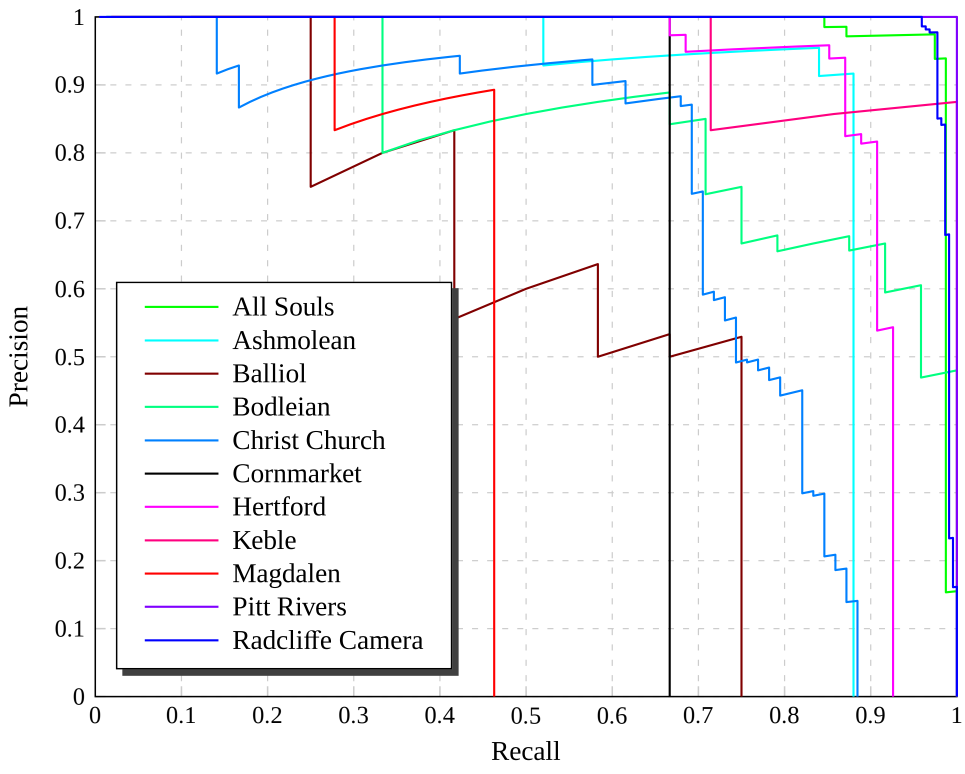
$$\text{recall} = \frac{|\{\text{positive retrived images}\}|}{|\{\text{positive images}\}|}. \tag{4.6}$$

Here, the images labeled as *Good* and *OK* are treated as positive images while images where the landmark is not present are treated as negative images. Images labeled as *Junk* are completely ignored and do not affect the precision-recall curve.

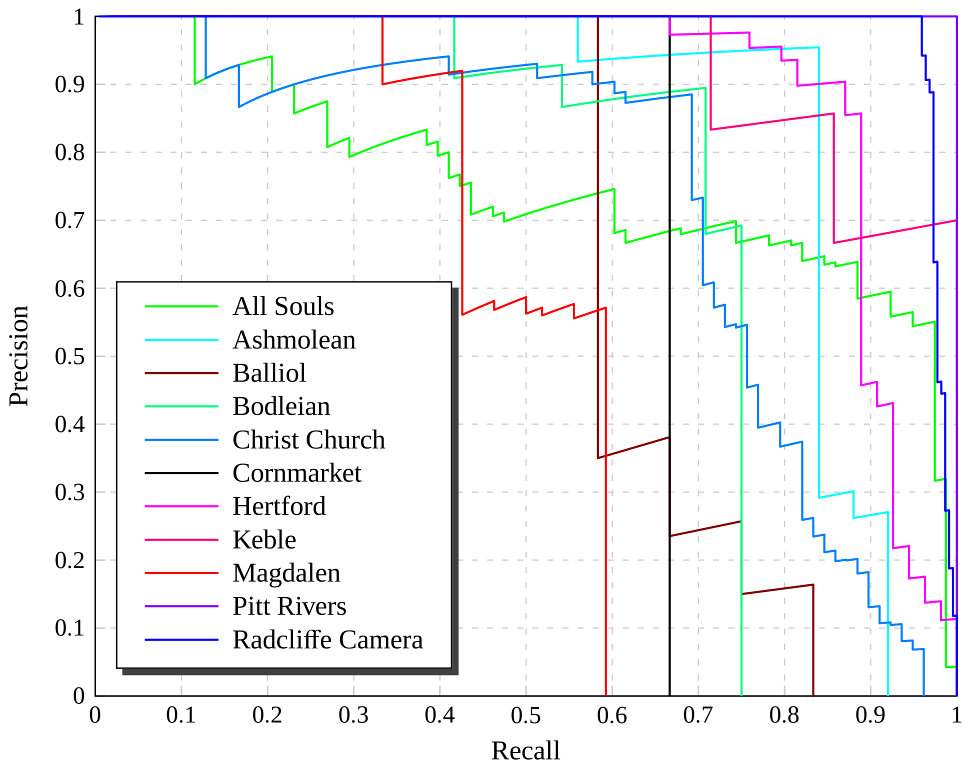
When a positive image is encountered, both precision and recall increase. By contrast, when a negative image is encountered, the precision decreases and the recall remains the same. This gives the precision-recall curve a tooth-like shape. The precision-recall curve of each ground truth landmark for MHOD with and without pruning are shown in Fig.4.11. To observe more clearly the overall performance, we compute the mean of precisions over all recall rates. Figure 4.12 illustrates the mean of precisions curve for MHOD with and without supervision.

To compare our results with other methods, we score the ranked image lists with the average precision (AP)[2]. The AP ranges from 0 to 1 and is given by the area under the precision-recall

[2]The AP is typically used in ranked lists because it takes into account the position of the relevant results.



(a)



(b)

Figure 4.11: Precision-recall of the 11 ground truth object: (a) with pruning and (b) without pruning.

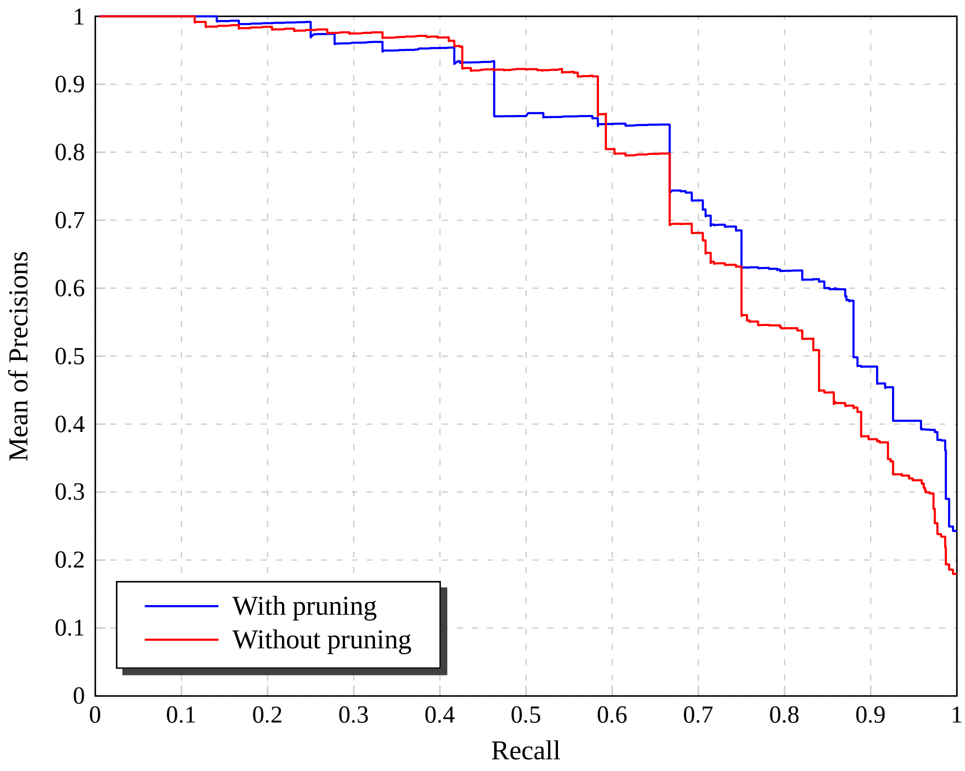


Figure 4.12: Mean of precisions curve for MHOD with pruning and without pruning.

curve. When AP = 1 the precision-recall curve becomes ideal, namely a precision 1 over all recall rates. For the sake of comparison, we follow the same approach of [43] and [44]. First, for each discovered object, the AP with respect to the ground truth landmark is computed from the ranked image list. Then, for each ground truth landmark, the discovered object with the highest AP is selected.

Table 4.2 shows the highest APs for LDA [44], gLDA [44], spectral clustering [43][3], and MHOD with and without pruning co-occurring word sets. Note that MHOD obtained the best results for all the landmarks (with the exception of Pitt Rivers for which all the methods obtained a perfect score) and in many cases with a substantial difference over the other three methods. This is clearly reflected on the average of the highest APs, where MHOD obtained a significantly better result.

Table 4.2 also presents the rank of the discovered objects that achieved the highest APs for MHOD. It is noteworthy that the discovered object that achieved the highest AP always had the

[3]The method described in [43] does not explicitly generate an object model, it only clusters images of the same object.

Table 4.2: Highest APs for LDA, gLDA, spectral clustering (SC) and MHOD with and without pruning (*).

Ground truth Landmark	LDA [44]	gLDA [44]	SC [43]	MHOD (*)	MHOD	Rank	Rank (*)
All Souls	0.90	0.95	0.93	0.75	**0.98**	2	2
Ashmolean	0.49	0.59	0.62	0.84	**0.85**	67	26
Balliol	0.23	0.23	0.33	**0.64**	0.56	208	73
Bodleian	0.51	0.64	0.61	0.70	**0.83**	474	4
Christ Church	0.45	0.60	0.67	0.72	**0.72**	3	3
Cornmarket	0.41	0.41	0.65	**0.66**	**0.66**	362	108
Hertford	0.64	0.65	0.70	**0.90**	**0.90**	4	5
Keble	0.57	0.57	0.93	0.93	**0.95**	69	43
Magdalen	0.20	0.20	0.20	**0.51**	0.43	328	56
Pitt Rivers	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	70	39
Radcliffe Camera	0.82	0.91	0.97	**0.98**	**0.98**	1	1
Average	0.56	0.61	0.69	0.78	**0.80**		

highest object rank among the objects associated with the same ground truth landmark. To see the effect of pruning co-occurring word sets in Sect. 4.1.2, this table also includes the results of MHOD without pruning co-occurring word sets. From Table 4.2, we can conclude that pruning co-occurring word sets improves the average of highest APs. This is because without pruning, meaningless object models can be derived from noisy co-occurring word sets.

Table 4.2 also shows the rank of the discovered object models that achieve the highest APs for MHOD. We confirmed visually that the object model that achieved the highest AP had the highest rank among the object models associated with the same landmark for any ground truth landmark. This fact also supports the meaningfulness of the object models discovered by MHOD.

To evaluate the sensitivity of MHOD to the parameters r and l of Min-Hashing in the co-occurring word set mining, we compute the average of highest APs for three different similarity thresholds ($s* = 0.16$, $s* = 0.19$ and $s* = 0.23$) with different values of r and l (see Table 4.3). In general, the results are stable for different all the cases. This shows that MHOD is robust to the choice of the parameter values r and l as well as the similarity threshold $s*$.

Finally, we investigate the sensitivity of MHOD to the parameter ϵ, which is the threshold

Table 4.3: Average of highest APs for $s* = 0.16$, $s* = 0.19$ and $s* = 0.23$ and different values of r and and l.

r	l	Avg. of highest APs	s*
	57	0.65	0.16
3	101	0.79	0.19
	169	**0.80**	0.23
	257	**0.80**	0.16
4	532	**0.80**	0.19
	1057	0.77	0.23
	1077	0.78	0.16
5	2799	**0.80**	0.19
	6610	0.69	0.23

for the overlap coefficient to merge co-occurring word sets in the agglomerative clustering of Sect. 4.1.3. Figure 4.13 illustrates the average of highest APs for different values of ϵ. Remarkably, MHOD performs stably for a wide range of ϵ from 0.33 to 0.99. Thus, we can say that MHOD is insensitive to the choice of ϵ.

4.4.3 Speed

All the experiments are carried out on a single 2.27GHz Intel Xeon PC with 4GB of memory. Table 4.4 summarizes the execution time for mining and clustering co-occurring word sets. Here, the execution time without pruning co-occurring word sets is also presented. Interestingly, pruning accelerates the speed of the object discovery. This is because pruning removes noisy and uninformative co-occurring word sets, shrinking the time for the clustering of co-occurring word sets. Without pruning, while a huge number of objects were discovered, many of them are meaningless and exploring the results may be cumbersome.

To demonstrate the scalability of MHOD, we apply it to a bigger dataset of 101,922 images which we call Rome100k. Rome100k was retrieved from Flickr using the keyword “Rome” as a query. We use the exact same setting as the Oxford buildings dataset also for the Rome100k dataset. The time for mining and clustering co-occurring word sets from the Rome100k[4] and

[4]Due to a memory issue, we had mistakenly reported that MHOD took 38.35 minutes for the Rome100k dataset. Although the actual time was slightly larger, it is still significantly lower than the other methods.

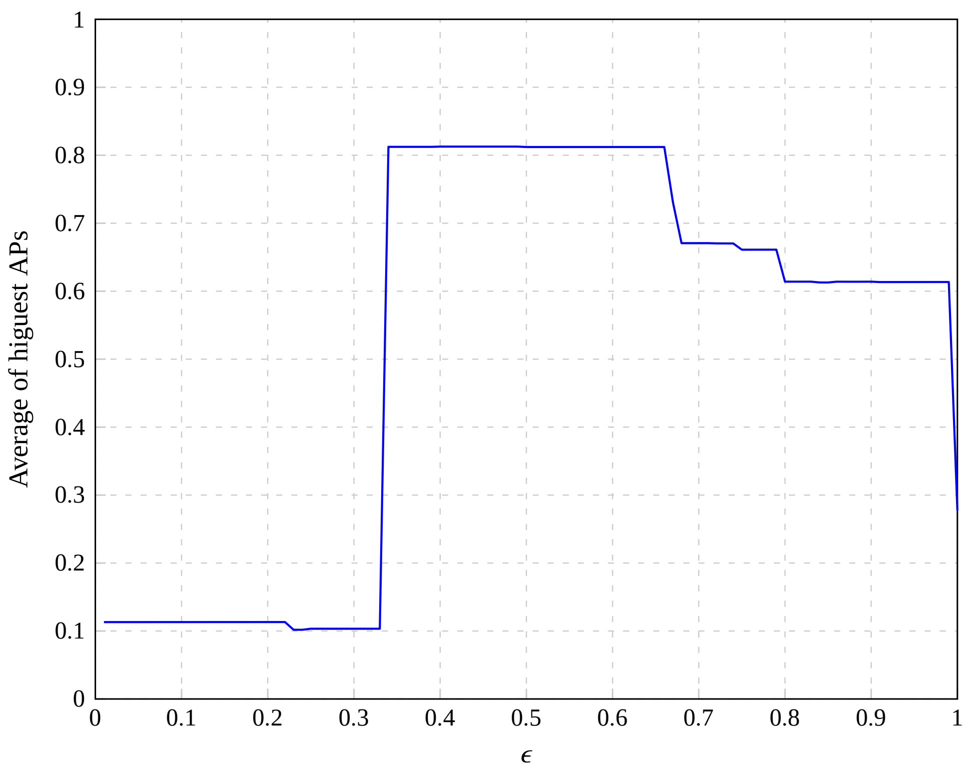


Figure 4.13: Average of the highest APs over different values of ϵ.

the Oxford datasets is presented in Table 4.5. Because the time for the Rome100k increased only slightly compared to that for the Oxford dataset, we can conclude that MHOD scales well with the number of images.

Table 4.5 also summarizes the processing time of other object discovery methods reported in the literature which were executed on various datasets and platforms. Though some literatures use PC clusters, it still takes much time to discover object. Because the platforms are different, it is difficult to compare the processing time between different methods. [44] reported that it took 2 hours on a dataset of 37,034 images to construct the matching graph [44] only on a single PC, while MHOD took 38.35 minutes on 101,922 images to derive the final object models on a single PC. We interpret this result as that MHOD is at least comparable to [44]. As for the memory consumption, for the Rome100k dataset, MHOD consumed at most 1200 MB.

In Fig. 4.14, the graphs of the execution time and space consumption of MHOD as the size of the image set grows large are presented. As we can observe, both the time and space complexity of MHOD grow linearly with the size of the image set, which proves the scalability of MHOD.

4.5 Discussion

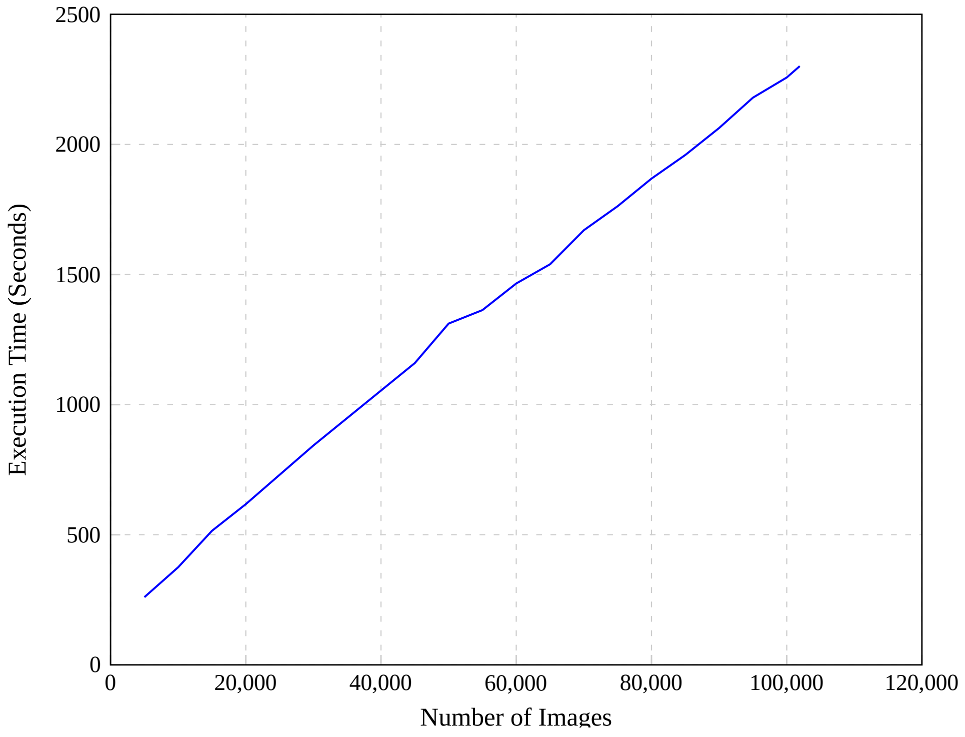
The quantitative results reported in the previous section show that MHOD outperforms LDA [44], gLDA [44] and spectral clustering [43] not only in efficiency but also in accuracy. A possible reason of the good performance of MHOD is that it only considers visual words that are highly discriminative, filtering out noisy visual words. In addition, in MHOD, once noisy visual words are filtered out, expressive object models are derived by clustering the highly discriminative visual words in an agglomerative manner. On the other hand, in [44] and [43], the number of different kinds of objects must be inferred by a sampling process. However, this strategy might not be so accurate, affecting the discovery results.

Table 4.4: Processing time for MHOD with and without pruning.

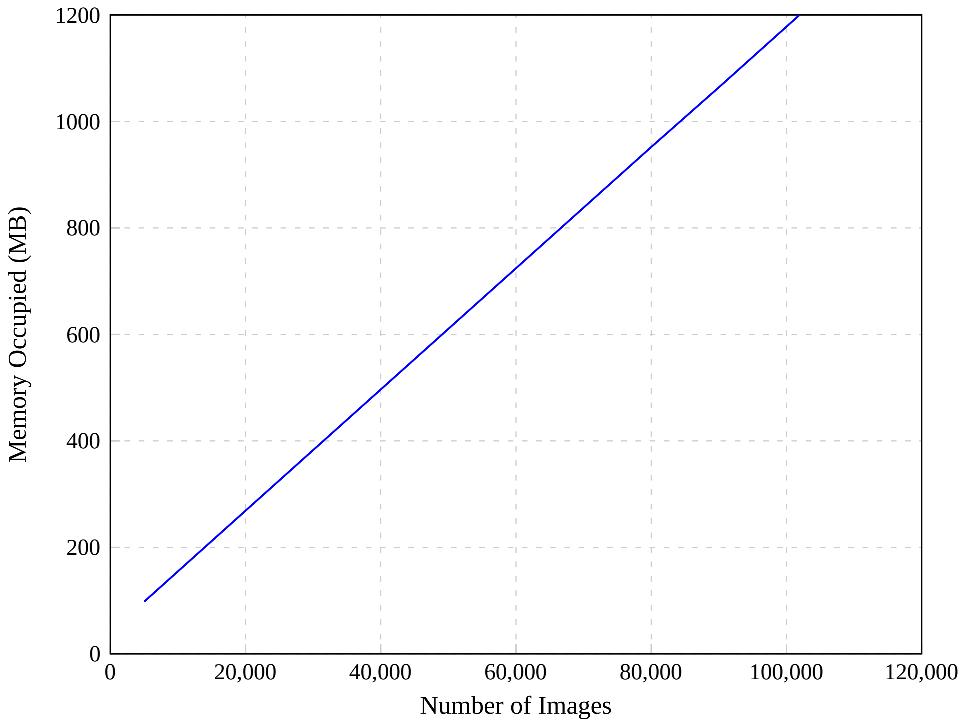
	Without pruning	With pruning
# of co-occurring word sets	950,730	287,927
# of discovered objects	649,876	33,102
Time for mining co-occurring word sets (secs)	288.110	288.110
Time for pruning (secs)	0	21.881
Time for clustering (secs)	191.695	75.508
Time for ranking images (secs)	6.092	2.676
Time for ranking objects (secs)	0.020	0.004
Total time (secs)	485.917	388.179

Table 4.5: Processing time for different methods

Method	Dataset	# of Images	# of Features	Platform	Time
[44]	Rome [54]	1,021,986	1,702,818,841	30 PCs	1 days
[44]	Statue of Liberty [54]	37,034	44,385,173	Single PC	2 hrs
SC [63]	Paris500k [6]	501,356	1,564,381,034	Multiple PCs	61.5 days
MHOD	Rome100k	101,922	460,894,893	Single PC	38.35 min
MHOD	Oxford [4]	5,062	16,334,970	Single PC	6.4 min



(a)



(b)

Figure 4.14: Execution time and space consumption of MHOD for different image set sizes.

Chapter 5

Conclusions

In this chapter, we summarize the contributions of the dissertation and discuss some promising directions for future works.

5.1 Summary

This dissertation advances over the problem of discovering particular objects from image collections without supervision. In particular, we showed that meaningful objects can be extracted by searching for reoccurring patterns in the image collection. Based on this strategy, we proposed two different methods. Both methods are implemented efficiently by using hashing techniques and therefore are suitable for large image collections.

5.1.1 Region-based method

- We proposed a method for discovering objects from segmented images.
- We demonstrated that it is possible to extract meaningful objects by mining frequent patterns of closely located image regions.
- In this method, different tasks are realized by using only hashing techniques, which simplifies its implementation.

- We incorporated two new strategies for judging similarity using hashing. Specifically, we extended the Euclidean LSH for similarity judgment considering the relative size and adapt standard hashing for matching with robustness to small variations.
- The proposed method can discover objects against simple background clutter, rotation, slide operations and small intra-class variations.
- Because objects must be isolated from the background, this method can not deal with occlusions and large amounts of clutter. In addition, the performance of this method is directly affected by the segmentation results.

5.1.2 Feature-based method

- We proposed a method which discovers object models by clustering features that consistently co-occur in a given image set.
- We showed that co-occurring features can be extracted efficiently by Min-Hashing.
- We demonstrated that despite disregarding the geometric relations between features, the generated models are highly discriminative and robust to occlusion, clutter and large variations of scale, illumination and viewpoint.
- The proposed method is scalable to huge image sets and large visual vocabularies as it performs the most demanding tasks by Min-Hashing.
- With this method, we could discover objects that correspond to human annotated objects from a benchmark dataset.
- This method discovered objects from a set of 101,922 images in just 38.35 minutes.
- The results obtained in a quantitative evaluation using ground truth data were superior to state-of-the-art methods both in efficiency and accuracy.

5.2 Future Works

Some possible extensions and directions for future work are discussed below.

- In this work, we have disregarded spatial relations between features. However, we believe that the incorporation of some kind of spatial information in the object models could improve their discrimination power.
- This work has focused on the discovery of particular objects. A possible future work would be to tackle the discovery of object categories by using the methods proposed in this dissertation.
- We have shown the efficiency of our feature-based method both in time and space using a single PC. Scaling up to multiple cores and computers represents an interesting future work.
- Although the methods proposed in this dissertation are applied to the discovery of objects from images, they can also be applied to other data mining problems which involve binary dyadic data such as topic discovery.

Bibliography

[1] `http://blog.flickr.net/en/2010/09/19/5000000000/`.

[2] `http://www.pixable.com/blog/2011/02/14/`.

[3] `http://blog.photobucket.com/blog/2010/12/p.html`.

[4] `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`.

[5] `http://www.flickr.com/`.

[6] `http://www.mmp.rwth-aachen.de/data/paris-dataset`.

[7] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *Proceedings of the 12th International Conference on Computer Vision*, pages 72–79, 2009.

[8] Alexandr Andoni and Piotr Indyk. Efficient algorithms for substring near neighbor problem. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1203–1212, 1994.

[9] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[10] Vassilis Athisos, Michalis Potamias, Panagiotis Papapetrou, and George Kollios. Nearest neighbor retrieval using distance-based hashing. In *Proceedings of the 24th IEEE International Conference on Data Engineering*, pages 327–336, 2008.

[11] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David M. Blei, and Michael I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[12] George Bebis, Michael Georgiopoulos, and Niels da Vitoria Lobo. Learning geometric hashing functions for model-based object recognition. In *Proceedings of the 5th International Conference on Computer Vision*, pages 543–548, 1995.

[13] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.

[14] Andrei Z. Broder. On the resemblance and containment of documents. *Computer*, 33(11):46–53, 2000.

[15] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380–388, 2002.

[16] Ondrej Chum and Jiri Matas. Large-scale discovery of spatially related images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:371–377, 2010.

[17] Ondrej Chum, James Philbin, Michael Isard, and Andrew Zisserman. Scalable near identical image and shot detection. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, pages 549–556, 2007.

[18] Ondrej Chum, James Philbin, and Andrew Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proceedings of the British Machine Vision Conference*, 2008.

[19] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):64–78, 2001.

[20] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*, pages 1–22, 2004.

[21] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pages 253–262, 2004.

[22] Jeff Edwards and Rahmat Shourschi. Recognition of multiple objects using geometric hashing techniques. In *Proceedings of the 32nd IEEE Conference on Decision and Control*, volume 2, pages 1617–1622, 1993.

[23] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[24] Rob Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, 2003.

[25] Claudio Gennaro, Pasquale Savino, and Pavel Zezula. Similarity search in metric databases through hashing. In *Proceedings of the ACM Workshops on Multimedia: Multimedia Information Retrieval*, pages 1–5, 2001.

[26] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–528, 1999.

[27] Taher H. Haveliwala, Aristides Gionis, and Piotr Indyk. Scalable techniques for clustering the web (extended abstract). In *Proceedings of the 3rd International Workshop on the Web and Databases*, pages 129–134, 2000.

[28] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, pages 177–196, 2001.

[29] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of 30th ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[30] Wolfson Haim J. and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science and Engineering*, 4(4):10–21, 1997.

[31] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Packing bag-of-features. In *Proceedings of the 12th International Conference on Computer Vision*, 2009.

[32] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, 2010.

[33] Joni-Kristian Kamarainen, Ville Kyrki, and Heikki Kalviainen. Local and global gabor features for object recognition. *Pattern Recognition and Image Analysis*, 17(1):93–105, 2007.

[34] Norbert Kruger. Object recognition with representations based on sparsified gabor wavelets used as local line detectors. In *Proceedings of the International Conference on Computer Analysis of Images and Patterns*, volume 1689 of *Lecture Notes in Computer Science*, pages 225–233. Springer-Verlag, 1999.

[35] Ville Kyrki and Joni-Kristian Kamarainen. Simple gabor feature space for invariant object recognition. *Pattern Recognition Letters*, 25(3):311–318, 2004.

[36] Xiaowei Li, Changchang Wu, Christopher Zach, Svetlana Lazebnik, and Jan-Michael Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *Proceedings of the 10th European Conference on Computer Vision*, pages 427–440, 2008.

[37] Nathan Linial, Eran London, and Rabinovich Yuri. The geometry of graphs and some of its algorithmic applications. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 577–591, 1994.

[38] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[39] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[40] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.

[41] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, 2006.

[42] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[43] James Philbin, Josef Sivic, and Andrew Zisserman. Object mining using a matching graph on very large image collections. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 738–745, 2008.

[44] James Philbin, Josef Sivic, and Andrew Zisserman. Geometric latent Dirichlet allocation on a matching graph for large-scale image datasets. *International Journal of Computer Vision*, pages 1–16, 2010.

[45] Till Quack, Bastian Leibe, and Luc J. Van Gool. World-scale mining of objects and events from community photo collections. In *Proceedings of the 7th ACM International Conference on Image and Video Retrieval*, pages 47–56, 2008.

[46] Khalid Saeed, Jerzy Pejas, and Romuald Mosdorf, editors. *Biometrics, Computer Security Systems and Artificial Intelligence Applications*, chapter Wavelet Transform in Face Recognition, pages 23–29. Springer US, 2006.

[47] Ruslan Salakhutdinov and Geoffrey Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, 2007.

[48] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[49] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their location in images. In *Proceedings of the 10th International Conference on Computer Vision*, pages 370–377, 2005.

[50] Josef Sivic, Bryan C. Russell, Andrew Zisserman, William T. Freeman, and Alexei A. Efros. Unsupervised discovery of visual object class hierarchies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[51] Josef Sivic and Andrew Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings of the 9th International Conference on Computer Vision*, pages 1470–1477, 2003.

[52] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *Proceedings of the 33rd International Conference and Exhibition on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pages 835–846, 2006.

[53] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, pages 189–210, 2008.

[54] Noah Snavely, Ian Simon, and Steven M. Seitz. Scene summarization for online image collections. In *Proceedings of the 11th International Conference on Computer Vision*, pages 1–8, 2007.

[55] Daniel L. Swets and John (Juyang) Weng. Using discriminate eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[56] Jiayu Tang and Paul H. Lewis. Non-negative matrix factorisation for object class discovery and image auto-annotation. In *Proceedings of the 7th ACM International Conference on Image and Video Retrieval*, pages 105–112, 2008.

[57] Yuan Yan Tang. *Wavelet theory and its application to pattern recognition*. World Scientific, 2000.

[58] Natsuki Tsunoda, Toshinori Watanabe, and Ken Sugawara. Image segmentation by adaptive thresholding of minimum spanning trees. *IEICE Transactions on Information and Systems (Japanese Edition)*, (2):586–594, 2004.

[59] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[60] Michel Vidal-Naquet and Shimon Ullman. Object recognition with informative features and linear classification. In *Proceedings of the 9th International Conference on Computer Vision*, volume 1, pages 281–288, 2003.

[61] Xiaogang Wang and Eric Grimson. Spatial latent dirichlet allocation. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1577–1584. 2008.

[62] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1761–1768. 2009.

[63] Tobias Weyand, Jan Hosang, and Bastian Leibe. An evaluation of two automatic landmark building discovery algorithms for city reconstruction. In *Proceedings of the Reconstruction and Modeling of Large-Scale 3D Virtual Environments*, 2010.

[64] Yan-Tao Zheng, Ming Zhao, Yang Song, Hartwig Adam, Ulrich Buddemeier, Alessandro Bissacco, Fernando Brucher, Tat-Seng Chua, and Hartmut Neven. Tour the world: building a web-scale landmark recognition engine. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2009.

List of Publications

Journals (Refereed)

1. Gibran Fuentes Pineda, Hisashi Koga and Toshinori Watanabe. Scalable Object Discovery: A Hash-based Approach to Clustering Co-occurring Visual Words. *IEICE Transactions on Information and Systems*. To appear in Vol. E94-D, No. 10, Oct. 2011. (Chapter 4)

International Conferences (Refereed)

1. Gibran Fuentes Pineda, Hisashi Koga and Toshinori Watanabe. Object Discovery by Clustering Correlated Visual Word Sets. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pages 750–753, 2010. (Chapter 4)

2. Gibran Fuentes Pineda, Hisashi Koga and Toshinori Watanabe. Unsupervised object discovery from images by mining local features using hashing. In *Proceedings of the 14th Iberoamerican Congress on Pattern Recognition (CIARP)*, volume 5856 of *Lecture Notes in Computer Science (LNCS)*, pages 978–985, 2009. (Chapter 3)

Author Biography

Gibran Fuentes Pineda received the B.E. degree in computer engineering and the M.S. degree in Microelectronics Engineering from the National Polytechnic Institute of Mexico. He is currently pursuing his Ph.D. degree in information systems at the University of Electro-Communications of Tokyo, Japan. His research interests lie in the areas of image data mining, image retrieval, pattern recognition, computer vision and machine learning.

Acknowledgements

This dissertation would not have been possible without the support and help of many people. First and foremost, I would like to thank my supervisors, professor Toshinori Watanabe and professor Hisashi Koga, for their guidance and encouragement and for all the stimulating discussions and many insightful ideas.

I gratefully acknowledge the members of the advisory committee for taking the time of revising my dissertation and for many valuable comments and suggestions that help to improve this dissertation.

I am profoundly grateful to my mother, my father, my sisters and all my family for their unconditional support through all these years. Especially, to my wife Mariana, who accepted a great deal of sacrifices to take this journey with me.

I feel really fortunate to have met many wonderful friends in Japan. I owe special thanks to the Minamoto and Yoshihara family for letting us be part of their family and for always taking care of us. I would also like to thank my dear friends Hayato and Toshiya for looking after us during our stay in Japan.

Last but not least, I thank all the members of the Watanabe & Koga Lab and the staff of UEC for kindly helping me every time that I needed.