博士論文

Extended Inverted Index and Fisher Kernel Approaches for Binary Local Feature-based Image Retrieval

(バイナリ局所特徴による画像検索のための
拡張転置インデックスおよびフィッシャーカーネル**)**

Yusuke Uchida
内田 祐介
Graduate School of Information Science and Technology
The University of Tokyo

This dissertation is submitted for the degree of
Doctor of Philosophy

June 2016

Acknowledgements

First and foremost, my deepest gratitude is to my advisor, Professor Shin'ichi Satoh, for his continuous support of my Ph.D. study. He always gives me positive and constructive comments. His insightful advices made my work more strict and generic.

I am also deeply grateful to my co-advisor, Professor Kiyoharu Aizawa and Associate Professor Toshihiko Yamasaki for the constructive discussions in advisory meetings. I would like to express my gratitude to Professor Yoichi Sato and Associate Professor Masashi Toyoda for their useful comments that helped me to refine my work.

Moreover, I would like to thank KDDI R&D Labs members. I would like to express my great appreciation to Dr. Shigeyuki Sakazawa for his many insightful comments and continuous encouragement. I am very grateful to Dr. Yasuyuki Nakajima and Dr. Hiromasa Yanagihara for supporting and encouraging this work.

Finally, I deeply thank my parents for their warm encouragement. I deeply grateful to my wife Yurika for her devoted support and encouragement. Without her support, I would not have been able to finish my work. I would also like to thank my sun Yuki for always giving me angelic smiles.

Abstract

With an increasingly wide-spread use of mobile devices such as Android phones or iPhones, mobile visual search (MVS) has become one of the major applications of image retrieval and recognition technology. With MVS, we can recognize the surrounding world with mobile devices using its built-in camera as an input to image recognition or retrieval systems. The recognition or retrieval results might be effectively shown using augmented reality technology on mobile devices for instance. Thus, mobile devices are now one of the best platforms for image retrieval systems.

While some research focuses on server-client systems in the context of MVS, we assume the situation that visual search is performed directly on the mobile device. We call the latter type of MVS "local MVS". Local MVS does not require any server and it works without a network, realizing faster recognition. In this thesis, we aim at developing a practical MVS system focusing on recent binary local features for efficiency. Although the performance of mobile devices has been improved, it is not sufficient to use non-binary features that requires a heavy processing load. Surprisingly, there are few studies focusing on the image retrieval using binary features. In many studies, binary features are used for feature-level matching between image pairs, not for image retrieval. One reason why binary feature-based image retrieval is not well-studied is that binary features are considered to be not enough robust to apply them to an image retrieval problem. However, we believe that binary features are very important for image retrieval on mobile devices and that there is a large room for improvement in the accuracy of binary feature-based image retrieval if we consider the recent significant advances in image representations for non-binary features. Thus, in this thesis, we explore the potential of binary features in the area of image retrieval and establish the basis of binary feature-based image retrieval that can be used to real applications. To this end, we propose and evaluate three approaches in order to achieve a practical binary feature-based image retrieval system.

First, we propose the application of the Fisher vector representation to binary features aiming at improving the accuracy of binary feature-based image retrieval

by considering underlying distribution of binary features. Main contribution of this approach is to model binary features using the Bernoulli mixture model (BMM) and derive the closed-form approximation of the Fisher vector of BMM. To the best of my knowledge, this is the first time to model binary features with BMM and apply the Fisher vector approach. We show that, by modeling binary features with BMM, it becomes possible to evaluate how informative different binary features are. Experimental results show that the proposed Fisher vector outperforms the BoVW method on various types of objects.

Second, we propose a substring extraction method that extracts informative bits from original binary vector and stores in the inverted index in order to improve the bag-of-visual words framework. These substrings are used to refine visual word-based matching. This is the first time to bring the idea of the Hamming embedding method to binary features. The advantage of this approach is its practicability. The developed system is very simple but effective, achieving good trade-offs between search precision, memory requirement, and speed. In addition, a modified version of the local naive Bayes nearest neighbor scoring method is proposed in the context of image retrieval, which considers the density of binary features in scoring each feature matching. The proposed system can retrieve the database with one million images in 87 [ms] and its accuracy significantly outperforms that of the state-of-the-art local MVS system.

Finally, we propose to integrate the advantages of the above two approaches. Starting with general match kernel, we show that the Fisher kernel-based similarity measurement can be implemented using the extended inverted index structure. Using the assumption that posterior probability is peaky, the Fisher kernel is linked with the BoVW framework, resulting in two proposed method, namely BMM-VW and BMM-FK. BMM-VW is a variant of BoVW, where VWs are defined by the BMM components. BMM-FK is the modified version of the second approach, where more appropriate similarity measurement is used. In order to ensure real-time applications, the method called randomized BMM trees is also proposed, which significantly accelerates the calculation of the quantization in BMM-VW and BMM-FK. In experiments, it is shown that the BMM-FK significantly outperforms the two previous approaches and the conventional state-of-the-art system in terms of the image retrieval accuracy.

We have developed real applications based on the above approaches, where a stand-alone system, a server-client system, and a hybrid system are used as a

backend. Through these practical applications, it has been proven that our developed systems have sufficient potential for practical usages.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

Image retrieval is the problem of searching for digital images in large databases. It can be classified into two types: text-based image retrieval and content-based image retrieval [1]. Text-based image retrieval (or concept-based image retrieval) refers to an image retrieval framework, where the images first are annotated manually, and then text-based Database Management Systems (DBMS) is utilized to perform retrieval [2]. However, the rapid increase of the size of image collection in the early 90's brought two difficulties. One is that the vast amount of labor is required in manually annotating the images. The other difficulty is the subjectivity of human perception; it sometimes happens that different people perceive the same image differently, resulting in different annotation results or different query keywords in search. This makes text-based image retrieval results less effective.

1.1.1 From Text-based to Content-based Image Retrieval

In order to overcome these difficulties, Content-Based Image Retrieval (CBIR) [3] or Query By Image Content (QBIC) [4] was proposed. In CBIR, images are automatically annotated with their own visual content by feature extraction process. The visual content includes colors [5–7], shapes [8], textures [9], or any other information that can be derived from the image itself. Extracted features representing visual content are indexed by high multi-dimensional indexing techniques to realize large-scale image retrieval [1].

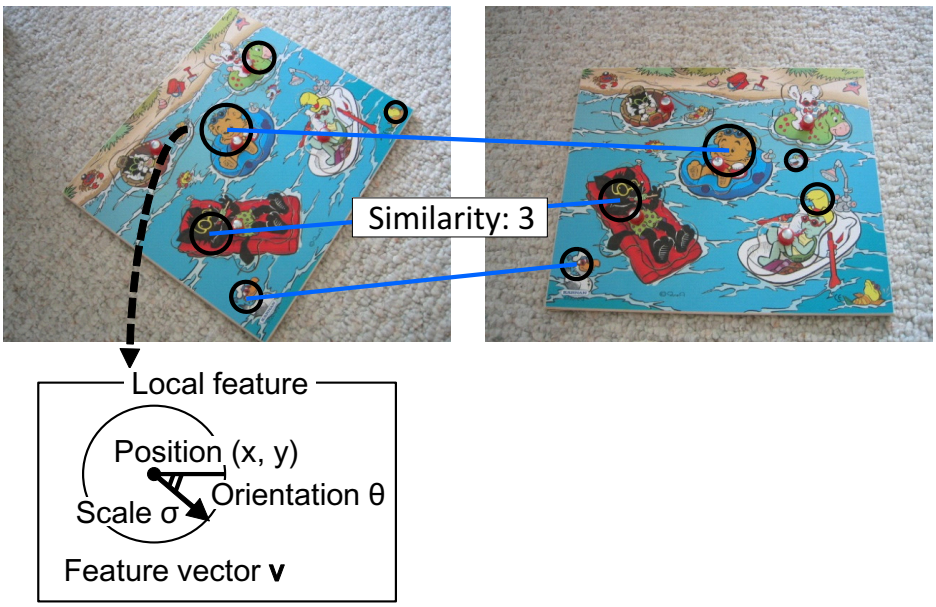


Figure 1.1: An example of local feature-based image retrieval.

1.1.2 Local Feature-based Image Retrieval

Although lots of image features are proposed in the middle of 90s in order to improve CBIR system, most of these features are *global* and therefore have difficulty in dealing with partial visibility and extraneous features. In order to handle partial visibility and transformations such as image rotation and scaling, a pioneer work on local feature-based image retrieval was done in [10]. In this framework, interest points are autoatically detected from an image, and then feature vectors are computed at the interest points. In search step, each of feature vectors extracted from a query image votes scores to matched referece features whose feature vectors are similar to the query feature vector. In local feature-based image retrieval, many local features are used in search, it is very robust against partial occulusion.

Figure 1.1 shows a toy example of local feature-based image retrieval, where the similarity of the two images is to be calculated. Fisrtly, local features are extracted from both images. Then, these local features are *matched* to generate pairs of local features whose feautre vectors are similar. In Figure 1.1, there are three pairs after matching. The most simple way to define the similarity between these two images is to use the number of the matched pairs: three in this case.

Table 1.1: The comparison of processing times [sec] of different local features run on a PC. The size of input images is 640x480. The default parameters defined in OpenCV 3.0 are used.

	Detection	Description	Total
SIFT	0.125	0.154	0.279
SURF	0.049	0.082	0.131
ORB	0.016	0.009	0.025

Table 1.2: The comparison of processing times [sec] of different local features run on a smartphone. The size of input images is 640x480. The default parameters defined in OpenCV 3.0 are used.

	Detection	Description	Total
SIFT	1.364	1.653	3.017
SURF	0.635	1.553	2.188
ORB	0.053	0.145	0.198

1.1.3 Advances of Mobile Devices

A smartphone is a mobile phone with an advanced mobile Operating System (OS) which combines features of a personal computer OS with other features useful for mobile or handheld use such as wireless communication or a built-in camera. While the history of the smartphone is old, recent major event in the history of mobile phones lead to a fundamental change was the launch of the first iPhone in 2007. In 2007, Apple Inc. introduced the iPhone, which is the first smartphones to use a multi-touch interface. In 2008, the first Android phone was released from HTC. Android is a mobile OS developed by Google Inc. These two platforms has driven the popularization of the smartphones. In terms of market size, it is reported that the global smartphone shipments for 2015 grew 10.3% year on year to 1.293 billion units[1].

1.1.4 Objective

With the above increasingly wide-spread use of mobile devices such as Android phones or iPhones, mobile visual search (MVS) has become one of the major applications of image retrieval and recognition technology. With MVS, we can recognize the surrounding world with mobile devices using its built-in camera as an input to image recognition or retrieval systems. The recognition or retrieval

[1]http://press.trendforce.com/press/20160114-2265.html

results might be effectively shown using augmetend reality technology. Thus, mobile devices are now one of the best platforms for image retrieval systems.

In this thesis, we develop a practical MVS system focusing on recent binary local features. While some research focuses on server-client systems in the context of MVS [11–15], the purpose of this thesis is to achieve fast and accurate recognition with lower memory requirements using only mobile devices [16]. We call the latter type of MVS "local MVS". Local MVS does not require any server and it works without a network, realizing faster recognition. The difficulty of local MVS lies in indexing of local features because it is necessary to fit the database to memory on mobile devices or an application size while maintaining retrieval accuracy. In other words, managing the trade-off between the memory size of the database and the accuracy of image retrieval is very important. Thus, the objective of this thesis is to achieve practical, accurate, low-memory, and fast recognition system working on mobile devices. The reason why we focus on binary local features is computational cost. Although the performance of mobile devices has been improved, it is not sufficient to use features that requires a heavy processing load. Table 1.1 and Table 1.2 compare processing times [sec] of different local features run on a PC and a smartphone respectively. A standard PC with a Core i7 2600 CPU 3.4GHz and Nexus7 (2013) with Qualcomm SnapDragon 600 (1.51GHz) are used here. We can see that the processing time on the mobile device is one order of magnitude longer than that on the PC. Furthermore, non-binary features (SIFT and SURF) are one order of magnitude slower than binary feature (ORB)[2]. Thus, non-binary features are not appropriate for MVS especially for real-time applications.

In spite of the above obervation that binary features are very important for applications on mobile devices, there are few studies focusing on the image retrieval using binary features [17, 18]. In many studies that propose binary features [19–23], the performance of (binary) features were evaluated in terms of the accuracy of feature-level matching for image pairs (e.g., precision-recall curve or matching repeatability), not evaluated in terms of the accuracy of image retrieval. Although several studies [19, 24] evaluated the accuracy of binary feature-based image retrieval, the framework used in these studies is too naive (nearest neighbor search + voting) and thus not practical. We think that one reason why binary feature-based image retrieval is not well-studied is that binary features are considered to be not enough robust to apply them to an image retrieval problem [16].

[2]The details of these local features will be described in Chapter 2.

We believe that binary features are very important for image retrieval, especially on mobile devices, when we consider their efficiency. In addition, we can say that there is a large room for improvement in the accuracy of binary feature-based image retrieval if we consider the recent significant advances in image representations for non-binary features. In this thesis, in order to clarify these issues, we explore the potential of binary features in the area of image retrieval and establish the basis of binary feature-based image retrieval that can be used to real applications.

1.2 Contribution of This Thesis

In this thesis, in order to realize practical mobile visual search system, we explore three approaches focusing on local binary features.

- **Fisher vectors for binary features.** We propose the application of the Fisher vector representation to binary features aiming at improving the accuracy of binary feature-based image retrieval by considering the distribution of binary features.Main contribution of this approach is to model binary features using the Bernoulli mixture model (BMM) and derive the closed-form approximation of the Fisher vector of BMM. To the best of my knowledge, this is the first time to model binary features with BMM and apply the Fisher vector approach. We show that, by modeling binary features with BMM, it becomes possible to evaluate how informative different binary features are. Experimental results show that the proposed Fisher vector outperforms the BoVW method on various types of objects. In addition, we also propose a fast approximation method to accelerate the computation of the proposed Fisher vectors by one order of magnitude with comparable performance.

- **Extended inverted index for binary features.** We propose a substring extraction method that extracts informative bits from original binary vector and stores in the inverted index in order to improve the bag-of-visual words framework. These substrings are used to refine visual word-based matching. This is the first time to bring the idea of the Hamming embedding method to binary features. The advantage of this approach is its practicability. The developed system is very simple but effective, achieving good trade-offs between search precision, memory requirement, and speed. In addition, a modified version of the local naive Bayes nearest neighbor scoring method is proposed in the context of image retrieval, which considers the density of binary features in scoring each

feature matching. Finally, in order to suppress false positives, we introduce a model check step after standard geometric verification using constraint on the configuration of a transformed reference image. The suppression of false positives is essential for real use cases. The proposed system can retrieve the database with one million images in 87 [ms] and its accuracy significantly outperforms that of the state-of-the-art local MVS system.

- **Linking Fisher kernel to inverted index-based systems.** In this approach, we propose to integrate the advantages of the above approaches. Starting with general match kernel, we show that the Fisher kernel-based similarity measurement can be implemented using the extended inverted index structure. Using the assumption that posterior probability is peaky, the Fisher kernel is linked with the BoVW framework, resulting in two proposed method, namely BMM-VW and BMM-FK. BMM-VW is a variant of BoVW, where VWs are defined by the BMM components. BMM-FK is the modified version of the second approach, where more appropriate similarity measurement is used. In order to ensure real-time applications, the method called randomized BMM trees is also proposed, which significantly accelerates the calculation of the quantization in BMM-VW and BMM-FK. In experiments, it is shown that the BMM-FK significantly outperforms the two previous approaches and the conventional state-of-the-art system in terms of the image retrieval accuracy.

Finally, we have developed real applications, which include a stand-alone system, a server-client system, and a hybrid system of them. Through these practical applications, it has been proven that our developed systems have sufficient potential for practical usages.

1.3 Structure of This Thesis

The structure of this thesis is presented in Figure 1.2. The background, the contribution, and the objective of the thesis is given in this chapter. In Chapter 2, we survey local features and image representations used in local feature-based image retrieval. In Chapter 3, we derive the Fisher vector representation suitable for binary features. In Chapter 4, we develop our local image retrieval system based on extended inverted index. In Cpapter 5, we combine the idea of the Fisher vector and the extended inverted index-based similarity search system. In Chapter 6, we

introduce real applications of our local feature-based image retrieval system. Finally, in Chapter 7, we conclude the thesis and present our future work.

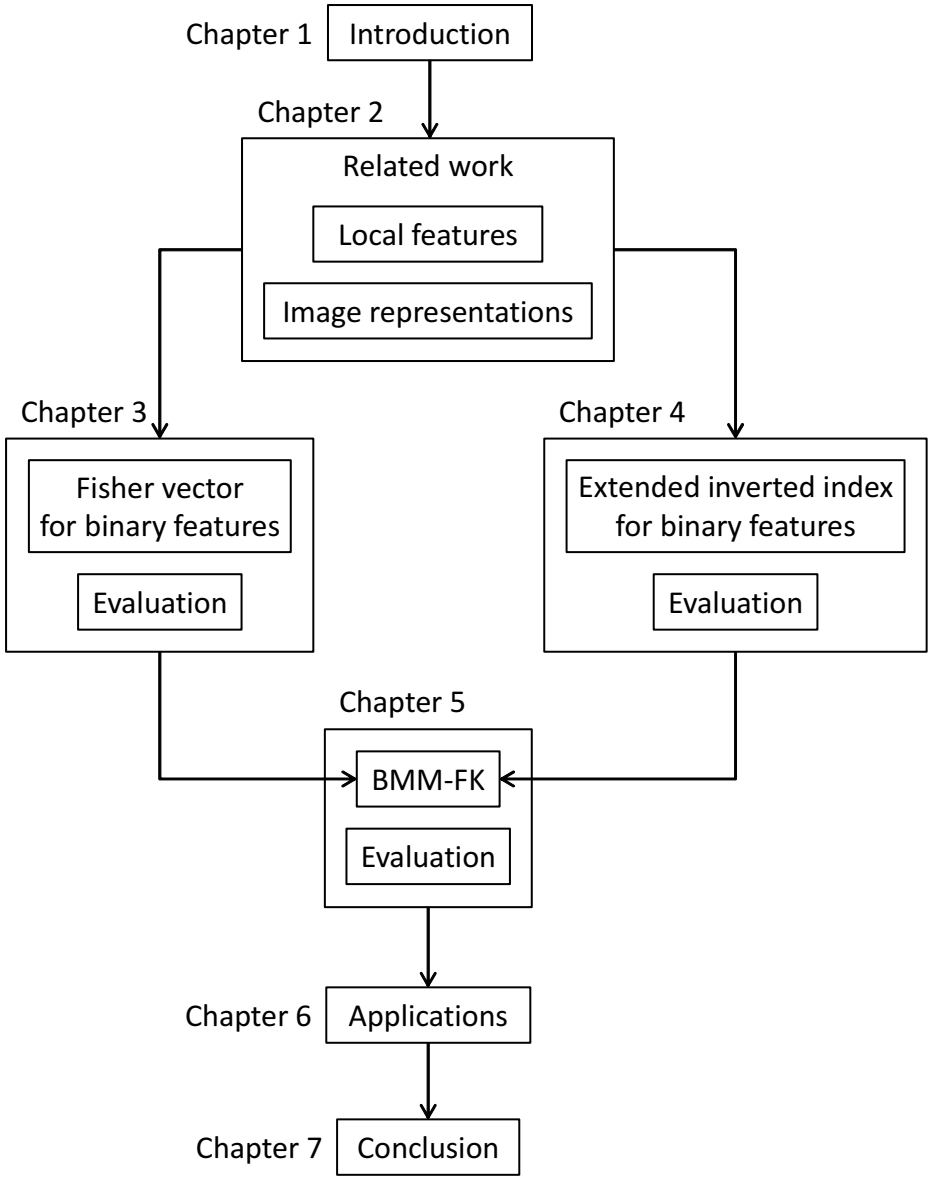


Figure 1.2: Structure of the thesis.

Chapter 2

Related Work

All of the local feature-based image retrieval system involves two important processes: local feature extraction and image representation. In local feature extraction, certain local features are extracted from an image. And then, in image representation, these local features are integrated or aggregated into a vector representation in order to calculate similarity between images[1].

In this chapter, we review related work in terms of local features and image representation [25]. In Section 2.1, various local features are introduced, which are the basis of recent local feature-based retrieval or recognition frameworks. In Section 2.2, various image representations are explained. In Figure 2.1, the overview of the history of local features and image representations.

2.1 Local Features

In this section, local features used in local feature-based image retrieval are reviewed. Local features are characterized by the combination of *feature detector* and *feature descriptor*. Feature detector finds feature points/locations, e.g. (x, y), or feature regions, e.g. (x, y, σ), where σ denotes the scale of the region. Feature descriptor extracts multi-dimensional feature vectors from the detected points or regions. While feature detectors and feature descriptors can be used in arbitrary combinations, specific combinations are usually used such as the DoG detector and the SIFT descriptor, multi-scale FAST detector and the BRIEF descriptor. In order to make local features invariant to rotation, the orientation of a local feature is estimated in many local features. In this thesis, the algorithms for the orientation estimation are

[1]Images are not necessarily represented by a single vector. Some methods simply defines the similarity between two sets of local features instead of explicitly integrating them into vectors.

included in the feature descriptor part, not in the detector part. While we focus on the systematic summary of local features, there are complementary comparative evaluations of local features [26–32, 24, 33, 34].

2.1.1 Feature Detectors

Feature detectors find multiple feature points or feature regions from an image. Feature detectors can be characterized by two factors: region type and invariance type. The region type represents the shape of a detected point or region such as corner or blob. The invariance type here represents to which transformations the detector is robust. The transformation can be a rotation, a similarity transformation, or an affine transformation. It is important to choose a feature detector with a specific invariance suitable for the problem we are solving.

Harris, Harris-Laplace, and Harris-Affine Detector

Harris detector [35] is one of the most famous corner detectors, which extends Moravec's corner detector. The original idea of the Moravec detector is to detect a pixel such that there is no nearby similar patch to the patch centered on the pixel. We assume a grayscale image I as an input. Let $I(u, v)$ denote the intensity of the pixel at (u, v) in I. Following the idea of Muravec detector, let $E(x, y)$ denote the weighted sum of squared differences caused by a shift (x, y):

$$E(x, y) = \sum_{u,v} w(u, v) \left(I(u + x, v + y) - I(u, v)\right)^2, \tag{2.1}$$

where $w(u, v)$ is a window function. The term $I(u + x, v + y)$ can be approximated by a Taylor expansion as

$$I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y, \tag{2.2}$$

where I_x and I_y denotes the partial derivatives of I with respect to x and y. Using this approximation, Eq. (2.1) can be written as

$$E(x, y) \approx \sum_{u,v} w(u, v) \left(I_x(u, v)x + I_y(u, v)y\right)^2. \tag{2.3}$$

This can be re-written in matrix form:

$$E(x, y) \approx (x, y)M(x, y)^\top, \tag{2.4}$$

where

$$M = \sum_{u,v} w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.5}$$

If $E(x, y)$ becomes large in any shift (x, y), it indicates a corner. This can be judged using the eigenvalues of M. Letting α and β denote the eigenvalues of M, $E(x, y)$ increases in all shift if both α and β are large. Instead of evaluating α and β directly, it is proposed to use $\mathrm{Tr}(M) = \alpha + \beta$ and $\mathrm{Det}(M) = \alpha\beta$ for efficiency [35]; the corner response R is defined as:

$$R = \mathrm{Det}(M) - k\,(\mathrm{Tr}(M))^2\,. \tag{2.6}$$

Thresholding on the values of R and performing non-maxima supporession, the Harris corners are detected from the input image.

The Harris detector is effective in the situation where scale change does not occur like tracking or stereo matching. However, as the Harris detector is very sensitive to changes in image scale, it is not appropriate for image retrieval, where the sizes of objects in query and reference images are frequently different. Therefore scale-invariant deature detector is essential for robust image recognition or retrieval.

Harris-Laplace detector [36] is a scale-adapted Harris detector. It firstly detects candidate feature points using the Harris detector on multiple scales (multi-scale Harris detector). Then, these candidate feature points are verified using the Laplacian to check whether the detected scale is maxima or not in the scale direction (cf. LoG Detector). The Harris-Laplace detector detects corner-like structures.

Harris-Affine detector [37, 38] is a affine-invariant feature detecctor. It firstly detects feature points using the Harris-Laplace detector. Then, iteratively refine these regions to affine regions using the second moment matrix as proposed in [39, 40]. The resulting Harris-Affine regions are characterized by ellipses.

Hessian, Hessian-Laplace, and Hessian-Affine Detector

Hessian detector [41] searches for image locations that have strong derivatives in two orthogonal directions. It is based on the matrix of second derivatives, namely

Hessian:

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}, \tag{2.7}$$

where $L(x, y, \sigma)$ is an image smoothed by a Gaussian kernel $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y). \tag{2.8}$$

The Hessian detector detects (x, y) as feature point such that the determinant of the Hessian H is local-maxima comapred with neighboring 8 pixels:

$$\det(H) = L_{xx}L_{yy} - L_{xy}^2. \tag{2.9}$$

Hessian-Laplace detector [36] is a scale-adapted Hessian detector. It firstly detects candidate feature points using the Hessian detector on multiple scales (multi-scale Hessian detector). Then, these candidate feature points are selected according to the Laplacian in the same way as Harris-Laplace. Note that the trade of the Hessian matrix is identical the Laplacian:

$$\mathrm{tr}(H) = L_{xx} + L_{yy}. \tag{2.10}$$

The Hessian-Laplace detector detects blob-like structures similar to the LoG or DoG detectors explained later. It is claimed that these methods often detect feature points on edges while the Hessian-Laplace does not, owing to the use of the determinant of the Hessian [27].

Hessian-Affine detector [27] is a affine-invariant feature detecctor and is imilar in spirt as the Harris-Affine detector. It firstly detects feature points using the Hessian-Laplace detector. Then, iteratively refine these regions to affine regions using the second moment matrix as done in the Harris-Affine detector.

LoG Detector

Detecting scale-invariant regions can be accomplished by searching for stable regions across all possible scales, using a continuous function of scale known as scale space. A scale space representation is defined by

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.11}$$

where $G(x, y, \sigma)$ is a Gaussian kernel:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-(x^2 + y^2)/2\sigma^2\right). \tag{2.12}$$

In [42], a blob detector that searches for scale space extrema of a scale-normalized Laplacian-of-Gaussian (LoG) $\sigma^2\nabla^2 L$, where

$$\nabla^2 L = L_{xx} + L_{yy}. \tag{2.13}$$

The term σ^2 is the normalization term, which normalizes response of LoG filter among different scales.

DoG Detector

Scale-Invariant Feature Transform (SIFT) [43, 44][2] is one of the most widely used local features due to its robustness. In detection of SIFT, it is proposed to use scale-space extrema in the Difference-of-Gaussian (DoG) function instead of LoG in order to efficiently detect stable keypoint. The DoG $D(x, y, \sigma)$ can be computed from the difference of two nearby scales separated by a constant multiplicative factor k:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{2.14}$$

$$= L(x, y, k\sigma) - L(x, y, \sigma). \tag{2.15}$$

The DoG is a close approximation to the scale-normalized LoG $\sigma^2\nabla^2 L$, which is shown using the heat diffusion equation:

$$\frac{\partial L}{\partial \sigma} = \sigma\nabla^2 L. \tag{2.16}$$

The term $\partial L/\partial\sigma$ can be approximated using the difference of nearby scales at $k\sigma$ and σ:

$$\frac{\partial L}{\partial \sigma} \approx \frac{L(x, y, k\sigma) - L(x, y, \sigma)}{k\sigma - \sigma}. \tag{2.17}$$

Thus, we get:

$$L(x, y, k\sigma) - L(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 L. \tag{2.18}$$

. The above equation shows that the response of DoG is already scale-normalized. Thus, DoG detector detects (x, y, σ) is a feature region if the response of $D(x, y, \sigma)$

[2]The SIFT algorithm includes both of detection and detection. In this thesis, they are distinguished by using the terms *the SIFT detector* and *the SIFT descriptor*.

is a local maxima or minima by comparing its 26 neighbors in terms of x, y, and σ dimensions. In [44], the DoG is efficiently calculated using image pyramid.

The detected region (x, y, σ) is further refined to sub-pixel and sub-scale accuracy by fitting a 3D quadratic to the scale-space Laplacian [45, 44]. After this refinement, detected regions are filterd out according to absolute values of their DoG responses and cornerness measures similar to the Harris detector in order to remove low contrast or edge regions [44].

SURF Detector

Speeded Up Robust Features (SURF) [46, 47] or fast Hessian detector is efficient approximation of the Hessian-Laplace detector. In [46, 47], it is proposed to approximate with box filters the Gaussian second-order partial derivatives L_{xx}, L_{xy}, and L_{yy}, which are required in the calculation of the determinant of the Hessian. These box filters can be efficiently calculated using *integral images* [48]. The SURF detector detects a scale-invariant blob-like features similar to the Hessian-Laplace detector. While the Hessian-Laplace detector uses the determinant of Hessian to select the location of the features and uses LoG to determine the characteristic scale, the SURF detecter uses the determinant of Hessian for both similar to the DoG detector.

FAST Detector

Most of the local binary features employ fast feature detectors. The Features from Accelerated Segment Test (FAST) [49–51] detector is one of such extremely efficient feature detectors. It can be considered as a simplied version of the Smallest Uni-value Segment Assimilating Nucleus Test (SUSAN) detector [52], which detects pixels such that there is few similar pixels around the pixels. The FAST Detector detects pixels that are brighter or darker than neighboring pixels based on the accelerated segment test as follows. For each pixel p, the intensities of 16 pixels on a Bresenham circle of radius 3 are compared with that of p, and are classified into three tyeps: *brighter*, *similar*, and *darker*. If there is least S connected pixels on the circle which are classified to brighter or darker, p is detected as a corner. In order to avoid detecting edges, S must be larger than nine and the FAST with $S = 9$ (FAST-9) is usually used.

In [49], it is proposed to accelerate this test by firstly checking the four pixels at the top, bottom, left, and right on the circle, achieving early rejection of the test. In [50], the segment test is further sped up by using a decision tree. By using a decision tree, the test is optimized to reject candidate pixels very quickly, realizing extremely

fast feature detection. In [19], it is proposed to filter out the detected FAST features according to their Harris scores. As the FAST detector is not scale-invariant, in order to ensure approximate scale invariance, feature points can be detected from an image pyramid [19], which is called the multi-scale FAST detector.

The AGAST descriptor [53], an acronym for Adaptive and Generic Accelerated Segment Test, is an extension of the FAST detector. There are two major improvements in the AGAST descriptor. The first one is the extension of the configuration space. In the AGAST descriptor, two additional types of the surrounding pixels are added in order to the configuration space: *not brighter* and *not darker*. By doing so, a more efficient decision tree can be constructed. The second improvement is that the AGAST descriptor adaptively switches two different decision trees according to the probability of a pixel state to be similar to the nucleus.

In [20], the multi-scale version of the AGAST detector is used. Local features are first detected from multiple scales, and then non-maxima suppresion is performed in scale-space according to the FAST score. Finally, scales and positoins of the detected local features are refined in a similar way to the SIFT detector.

2.1.2 Feature Descriptors

Differential Invariants Descriptor

Differential invariants descriptor was used in the pioneer work of local feature-based image retrieval [10]. It consists of components of local jets [54] and has rotation invariance:

$$v = \begin{bmatrix} L \\ L_i L_i \\ L_i L_{ij} L_j \\ L_{ii} \\ L_{ij} L_{ji} \\ \epsilon_{ij}(L_{jkl} L_i L_k L_l - L_{jkk} L_i L_l L_l \\ L_{iij} L_j L_k L_k - L_{ijk} L_i L_j L_k \\ -\epsilon_{ij} L_{jkl} L_i L_k L_l \\ L_{ijk} L_i L_j L_k \end{bmatrix}, \tag{2.19}$$

where $i, j, k, l \in \{x, y\}$, and L_x represents the convolution of image I with the Gaussian derivative G_x in terms of x direction.

One approach to attain rotation-invariant local features is to adopt a scale-invariant descriptor like this differential invariants descriptor. However, this ap-

proach results in less distinctive feature vector because it discards image information so that the resulting vector becomes the same irrespective of the degree of rotation. Therefore, many descriptors adopts an orientation estimation step, and then feature descriptors extracts (scale-variant) feature vecctors relative to this orientation and therefore achieve invariance.

SIFT Descriptor

The SIFT [43, 44] descriptor is one of the most widely used feature descriptors, and sometimes combined with the other detectors (e.g. the Harris/Hessian-Affine detectors) as well as the SIFT detector. In the SIFT descriptor, the orientation of local region (x, y, σ) is estimated before description as follows. Firstly, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are computed using pixel differences:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}, \tag{2.20}$$

$$\theta(x, y) = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}, \tag{2.21}$$

where $L(x, y)$ denotes the intensity at (x, y) in the image I smoothed by the Gaussian with the scale parameter corresponding to the detected region. Then, an orientation histogram is formed from the gradient orientations of sample pixels within the feature region; the orientation histogram has 36 bins covering the 360 degree range of orientations. Each pixel votes a score of the gradient magnitude $m(x, y)$ weighted by a Gaussian window to the bin corresponding to orientation $\theta(x, y)$. The highest peak in the histogram is detected, which corresponds to the dominant direction of local gradients. If any, the other local peaks that are within 80% of the highest peak are used to create local features with that orientations [44].

After the assignment of the orientation, the SIFT descriptors are computed for normalized image patches. The descriptor is represented by a 3D histogram of gradient location and orientation, where location is quantized into a 4×4 location grid and the orientation is quantized into eight bins, resulting in the 128-dimensional descriptor. For each of sample pixels, the gradient magnitude $m(x, y)$ weighted by a Gaussian window is voted to the bin corresponding to (x, y) and $\theta(x, y)$ similar to the orientation estimation. In order to handle a small shift, a soft voting is adopted, where scores weighted by trilinear interpolation are additionally voted to seven neighbor bins (voted to eight bins in total). Finally, the feature vector is ℓ_2 normalized to reduce the effects of illumination changes.

It is shown that certain post processing improves the discriminative power of the SIFT descriptor [55, 56]. In [55], it is proposed to transform the SIFT descriptors by (1) ℓ_1-normalization of the SIFT descriptor instead of ℓ_2 and (2) taking square root each dimension. The resulting descriptor is called RootSIFT. Comparing RootSIFT using ell_2 distance correspond to using the Hellinger kernel in comparing the original SIFT descriptors. In [56], explict feature map of the Dirichlet Fisher kernel is proposed to transform the histogram-based feature vector (including the SIFT descriptor) to more discriminative one.

SURF Descriptor

The orientation assignment of the SURF Descriptor [46, 47] is similar to the SIFT descriptor. While the gradient magnitude and orientation are calculated from the image smoothed by the Gaussian in the SIFT descriptor, the Haar-wavelet responses in x and y directions are used in the SURF descriptor, where integral images are used for efficient calculation of the Haar-wavelet response. Letting s denote the characteristic scale of the SURF feature, the size of the Haar-wavelet is set to $4s$. The Haar-wavelet responses of the pixels in a circular with the radius of $6s$ are accumulated using a sliding window with the size of $\pi/3$ and the dominant orientation is obtained.

In description, the feature region is first rotated using the estimated orientation, and divided into 4×4 subregions. For each of the subregions, d_x, d_y, $|d_x|$, and $|d_y|$ are computed at 5×5 regularly spaced sample points, where d_x and d_y are the Haar-wavelet responses with the size of $2s$ in x and y directions. These values are accumulated with the Gaussian weights, resulting in a subvector $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. The subvectors of 4×4 regions are concatenated to form the 64-dimensional SURF descriptor.

BRIEF Descriptor

The Binary Robust Independent Elementary Features (BRIEF) descriptor [57] is a pioneering work in the area of recent binary descriptors [30]. Binary descriptors are quite different from the descriptors discussed above because they extract binary strings from patches of interest regions for efficiency instead of extracting gradient-based high-dimensional feature vectors like SIFT. The distance calculations between binary features can be done efficiently by XOR and POPCNT operations.

The BRIEF descriptor is a bit string description of an image patch constructed from a set of binary intensity tests. Many binary descriptors utilize similar binary tests in extracting binary strings. Consider the t-th smoothed image patch p_t, a binary test τ for d-th bit is defined by:

$$x_{td} = \tau(p_t; a_d, b_d) = \begin{cases} 1 & \text{if } p_t(a_d) \geq p_t(b_d) \\ 0 & \text{else} \end{cases}, \tag{2.22}$$

where a_d and b_d denote relative positions in the patch p_t, and $p_t(\cdot)$ denotes the intensity at the point. Using D independent tests, we obtain D-bit binary string $x_t = (x_{t1}, \cdots, x_{td}, \cdots, x_{tD})$ for the patch p_t. In the original BRIEF descriptor [57], the relative positions $\{(a_d, b_d)\}_d$ are randomly selected from certain probabilistic distributions (e.g. Gaussian).

The ORB descriptor [19] is a modified version of the BRIEF descriptor, where two improvements are proposed: a orientation assignment and a learning method to optimize the positions $\{(a_d, b_d)\}_d$. In the orientation assignment, the intensity centroid [58] is used. The intensity centroid C is defined as

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right), \tag{2.23}$$

where m_{pq} is the moments of the feature region:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y). \tag{2.24}$$

The orientation θ of the vector from the center of the feature region to C is obtained as:

$$\theta = \text{atan2}(m_{01}, m_{10}). \tag{2.25}$$

In athe learning method, the positions $\{(a_d, b_d)\}_d$ are optimized so that the average value of each resulting bit is close to 0.5, and bits are not correlated. This is achieved by an algorithm which greedy chooses the best binary test from all possive tests:

1. Calculate means of bits of all binary tests using traning patches rotated by θ.

2. Sort the bits according to their distance from a mean of 0.5. Let T denote the resulting vector.

3. Initialize the result vector R by the first test in T and remove it from T.

4. Take the next test from T, and compare it against all tests in R. If its absolute correlation is smaller than a threshold, discard it; else add it to R. Repeat this step until there are 256 tests in R. If there are fewer than 256, raise the threshold and try again.

This descriptor is usually combined with the multi-scale FAST detector, and therefore coined as Oriented FAST and Rotated BRIEF (ORB).

The BRISK descriptor [20], an acronym for Binary Robust Invariant Scalable Keypoints, is a binary fescriptor similar to the ORB descriptor but proposed at the same time. The major difference from the ORB descriptor is that the BRISK descriptor utilizes different sampling patterns for binary tests. The BRISK sampling pattern is defined by the locations $p_i = (a_i, b_i)$ equally spaced on circles concentric with the keypoint, similar to the DAISY[59, 60] descriptor. For each location, the responses of different sizes of Gaussian kernel $I(p_j, \sigma)$ and $I(p_i, \sigma_i)$ at two different points p_i and p_j are compared in the tests as done in Eq. (2.22), while the intensities at two different points of smoothed image are compared in the ORB descriptor. In the BRISK descriptor, sampling-point pairs whose distances are shorter than a threshold are used for description, resulting in the 512-bit descriptor. The orientation $\tan^{-1}(g)$ is also estimated using the sampling pattern:

$$g(p_i, p_j) = (p_i - p_j)\frac{I(p_j, \sigma_j) - I(p_i, \sigma_i)}{\|p_j - p_i\|^2}, \tag{2.26}$$

$$g = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \frac{1}{L} \sum_{(p_i, p_j) \in \mathcal{L}} g(p_i, p_j), \tag{2.27}$$

where $\mathcal{L}$ is sampling-point pairs whose distances are relatively long.

The FREAK descriptor [21], an acronym for Fast REtinA Keypoint, is a binary fescriptor similar to the ORB and FREAK descriptor. The FREAK sampling pattern mimics the *retinal ganglion cells* distribution with their corresponding receptive fields, resulting in the very similar sampling pattern fo the DAISY[59, 60] descriptor. In the FREAK descriptor, the responses of different sizes of Gaussian kernel at two different points are compared in the tests similar to the BRISK descriptor, but the learning method in the ORB descriptor is used to select the effective binary tests. The orientation is also calculated in a similar way to the BRISK descriptor. The differences are the sampling pattern and the number of point pairs. The number of point pairs is reduced to 45, achieving smaller memory requirement.

2.2 Image Representations

2.2.1 Bag-of-Visual Words

The BoVW framework is the de-facto standard way to encode local features into a fixed length vector. The BoVW framework is firstly proposed in the context of object matching in videos [61], it has been used in various tasks in image retrieval [62–64], image classification [65–67], and video copy detection tasks [68, 69]. In the BoVW framework, represantative vectors called visual words (VWs) or visual vocabulary are created. These representative vectors are usually created by applying k-means algorithm to training vectors, and resulting centroids are used as VWs. Feature vectors extracted from an image are quantized into VWs, resulting in a histogram representation of VWs. Image (dis)similarity is measured by ℓ_1 or ℓ_2 distance between the normalized histograms.

As the histograms are generally sparse[3], an inverted index and a voting function enables an efficient similarity search [61]. Figure 2.2 shows a framework of image retrieval using the inverted index data structure. The inverted index contains a list of containers for each VW, which store information of reference features such as the identifiers of reference images, the positions (x, y) of the reference features, or other information used in search step.

The framework involves three steps: training, indexing, and search steps. In training step, VWs are trained by performing the k-means algorithm to training vectors. Other trainings requried for indexing or search are done, if any. In indexing step, feature regions are firstly detected in a reference image, and then feature vectors are extracted to describe these regions. Finally, each of these reference feature vectors is quantized into VW, and the identifier of the reference image is stored in the corresponding lists with other metadata related the reference feature. In search step, feature regions are detected in a query image, feature vectors are extracted, and these query feature vectors are quantized into VWs in the same manner as done in the indexing step. Then, each of query feature vote a certain score to reference images whose identifiers are found in the corresponding lists. A Term Frequency-Inverse Document Frequency (TF-IDF) scoring [61] is often used in voting function. The voting scores are accumulated over all of the query feature

[3]Note that, in image classification tasks, the BoVW histogram is often not sparse but dense because extremely larger number of features are extracted with dense grid sampling and the number of VWs is relatively small. Therefore, it is not standard to use inverted index but simply treat BoVW histogram as a dense vector.

features, resulting similarities between the query image and the reference images. The results obtained in voting function optionally refined by Geometric Verification (GV) or spatial re-ranking [63, 70], which will be described later.

The BoVW is the most widely used framework in local feature-based image retrieval, and therefore many extensions of the BoVW framework are proposed. In the following, we comprehensively review these extensions. We classify the BoVW extensions into the following groups in this thesis: large vocabulary, multiple assignment, post-filtering, weighted voting, geometric verification, weak geometric consistency, and query expansion. These are reviewed one by one.

Large Vocabulary

Using a large vocabulary in quantization, e.g. one million VWs, increases discriminative power of VWs, and thus improves search precision. In [62], it is proposed to quantize feature vectors using a vocabulary tree, which is created by hierarchical k-means clustering (HKM) instead of a flat k-means clustering. The vocabulary tree enables extremely efficient indexing and retrieval while increasing discriminative power of VWs. A hierarchical TF-IDF scoring is also proposed to alleviate quantization error caused in using a large vocabulary. This hierarchical scoring can be considered as a kind of multiple assignment explained later.

In [63], approximate k-means (AKM) is proposed to create large vocabulary. AKM is an approximated version of k-means algorithm, where an approximate nearest neighbor search method is used in assigning training vectors to their nearest centroids. In AKM, a forest of randomized k-d trees [71–73] is used for approximate nearest neighbor search, where are the randomized k-d trees are simultaneously searched using a single priority queue in a best-bin-first manner [74]. This nearest neighbor search is performed in quantization as well as in clustering. It is shown that AKM outperforms HKM in terms of image search precision. This is because HKM minimizes quantization error only locally at each node while the flat k-means minimizes total quantization error, and AKM successfully approximate the flat k-means clustering.

In [75, 76], the combination of the above HKM and AKM, namely approximate hierarchical k-means (AHKM), is proposed to construct further larger vocabulary. The AHKM tree consists of two levels, where each level has 4K nodes. The first level is constructed by AKM using randomly sampled training vectors. Then, over 10 billion training vectors are devided into 4K clusters by using the first level centroids. For each of the above 4K clusters AKM is further applied to construct the second

level with 4K centroids, resulting in 16M VWs. In the construction of the first level, th tree structure is balanced so that averaging the speed of the retrieval [77, 78].

Multiple Assignment

One significant drawback of VW-based matching is that two features are matched if and only if they are assigned to the same VW. Figure 2.3 illustrates this drawback. In Figure 2.3 (a), two features f_1 and f_2 extracted from the same object are close to each other in the feature vector space. However, there are the boundary of the Voronoi cells defined by VWs, and they are assigned to the different VWs v_1 and v_j. Therfore, f_1 and f_2 are not matched in the naive BoVW framework. Multiple assignment (or soft assignmet) is proposed to solve this problem. The basic idea is to assign feature vectors not only to the nearest VW but to the several nearest VWs. Figure 2.3 (b) explains how it works. Suppose f_1 is a query vector and assigned to the nearest two VWs, v_i and v_j. In this case, reference features inclusing f_2 in the gray area are matched to f_1. In general, multiple assignment improves recall of matching features while degrading precision because each feature is matched with larger number of features in the database compared with hard (single) assignment case.

In [79], each of reference features is assigned to the fixed number r of the nearest VWs in indexing and the corresponding score $\exp -\frac{d^2}{2\alpha^2}$ is additionally stored in the inverted index, where d is the distance from the VW to the reference feature, and α is a scaling parameter. It is shown that the multiple assignment brings a considerable performance boost over *hard-assignment* [63]. This multiple assignment is called *reference-side* multiple assignment because it is done in indexing. Beucase reference-side multiple assignment increases the size of the index almost proportionally to the factor r, the following *query-side* multiple assignment is often used. In [79], it is also proposed to perform multiple assignment against image patch, namely *image-space* multiple-assignment. In *image-space* multiple-assignment, a set of descriptors is extracted from each image patch by synthesizing deformations of the patch in the image space and assign each descriptor to the nearest visual word. However, it is shown that, compared to descriptor-space soft-assignment explained above, image-space multiple assignment is much more computationally expensive while not so effective.

In [77], it is proposed to perform multiple assignment to a query feature (query-side multiple assignment), where the distance d_0 to the nearest VW from a query feature is used to determine the number of multiple assignments. The query feature is assigned to the nearest VWs such that the distance to the VW is smaller than αd_0

(α = 1.2 in [77]). This approach adaptively changes the number of assigned VWs according to ambiguity of the feature.

While all of the above methods utilize the Euclidean distance in selecting VWs to be assigned, in [75, 76], it is proposed to exploit a probabilistic relationships $P(W_j|W_q)$ of VWs in multiple assignment. $P(W_j|W_q)$ is the probability of observing VW W_j in a reference image when VW W_q was observed in the query image. In other words, $P(W_j|W_q)$ represents which other VWs (called alternative VWs) that are likely to contain descriptors of matching features. The probability is learnt from a large number of matching image patches. For each VW W_q, a fixed number of alternative VWs that have the highest conditional probability $P(W_j|W_q)$ is recorded in a list and used in multiple assignment; a query feature assigned to the VW W_q, it is also assinged to the VWs in the list.

Post-filtering

As the naive BoVW framework suffers from many false matches of local features, post-filtering approaches are proposed to eliminate unreliable feature matches. In post-filtering approaches, after VW-based matching, matched feature pairs are further filtered out according to the distances between them. For example, in Figure 2.3, two features f_1 and f_3 are far from each other in feature vector space but in same the Voronoi cell, thus they are matched in the naive BoVW framework. In Figure 2.3 (c), post-filtering is applied; the query feature is matched with only the reference features in the gray area, flitering out the feature f_3. Post-filtering approches have similar effect as using a large vocabulary because both of them improve accuracy of feature matching. While post-filtering approaches try to improve the precision of feature matches with only slight degradation of recall, simply using a large vocabulary causes a considerable degradation of recall in feature matching [64].

In post-filtering approaches, after VW-based matching, distances between a query feature and reference features that are assigned to the same VW should be calculated for post-filtering. However, as exact distance calculation is undesirable in terms of computational cost and memory requirement to store raw feature vectors. Therefore, in [64, 77, 80, 81], feature vectors extracted from reference images are encoded into binary codes (typically 32-128 bit codes) via random orthogonal projection followed by thresholding for binarizing projected vectors. While all VWs share a single random orthogonal matrix, each VW has individual thresholds so that feature vectors are binarized into 0 or 1 with the same probability. These codes are stored in an inverted index with image identifiers (sometimes with other information on the

features. In a search step, after VW-based matching, Hamming distances between codes of query and matched reference features are calculated. Matched features with larger Hamming than a predefined threshold are filtered out, which considerably improves the precision of matching with only slight degradation of recall.

In [82–84], a product quantization-based method is proposed and shown to outperform other short codes like spectral hashing (SH) [85] or a transform coding-based method [86] in terms of the trade-off between code length and accuracy in approximate nearest neighbor search. In the PQ method, a reference feature vector is decomposed into low-dimensional subvectors. Subsequently, these subvectors are quantized separately into a short code, which is composed of corresponding centroid indices. The distance between a query vector and a reference vector is approximated by the distance between a query vector and the short code of a reference vector. Distance calculation is efficiently performed with a lookup table. Note that the PQ method directly approximates the Euclidean distance between a query and reference vector, while the Hamming distance obtained by the HE method only reflects their similarity.

Weighted Voting

In voting function, TF-IDF scoring [61] is often used. Some researches try to improve the image retrieval accuracy by modifying this scoring. One directon to do this is the modification of the standard IDF. In [87, 88], ℓ_p-norm IDF is proposed, which can be considered as a generalized version of the standard IDF. The standard IDF weight for the visual word z_k is defined as:

$$IDF(z_k) = \log \frac{N}{n_k}, \tag{2.28}$$

where N denotes the number of images in the database and n_k denotes the number of images that contain z_k. The ℓ_p-norm IDF is defined as:

$$pIDF(z_k) = \log \frac{N}{\sum_{I_i \in P_k} w_{i,k} v_{i,k}^p}, \tag{2.29}$$

where $v_{i,k}$ denotes the occurrences of z_k in the image I_i and $w_{i,k}$ is normalization term. It is reported that when p is about 3-4, ℓ_p-norm IDF achieves better accuracy than the standard IDF. In [89], BM25 with exponential IDF weights (EBM25) is proposed. BM25 is a ranking function used for document retrieval and it includes the IDF term in its definition. This IDF term is extended to the exponential IDF that is capable of

suppressing the effect of background features. In [90], theoretical scoring method is derived by formulating the image retrieval problem as a maximum-a-posteriori estimation. The derived score can be used an alternative to the standard IDF.

The distances between the query and reference features obtained in the post-filtering approach are aften exploited in weighted voting. In [91], the weight is calculated as a Gaussian function of a Hamming distance between the query and reference vector. In [77], the weight is calculated based on the Hamming distance between the query and reference vector and the probability mass function of the binomial distribution. In [92], the weight is calculated based on rank information because a rank criterion is used in post-filtering in the literature, while in [83], the weight is calculated based on ratio information.

Geometric Verification

Geometric Verification (GV) or spatial re-ranking is important step to improve the results obtained by voting function [63, 70]. In GV, transformations between the query image and the top-R reference images in the list of voting results are estimated, eliminating matching pairs which are not consistent with the estimated transformation. In the estimation, the RANdom SAmple Consensus (RANSAC) algorithm or its variants [93, 94] are used. Then, the score is updated counting only inlier pairs. As a transformation model, affine or homography matrix is usually used.

In [44] a 4 Degrees of Freedom (DoF) affine transformation is estimated in two stages. First, a Hough scheme estimates a transformation with 4 parameters; 2D location, scale, and orientation. Each pair of matching regions generates these parameters that *vote* to a 4D histogram. In the second stage, the sets of matches from a bin with at least 3 entries are used to estimate a finer 2D affine transform. In [63], three affine sub-groups for hypothesis generation are compared, with DoF ranging between 3 and 5. It is shown that 5 DoF outperforms the others but the improvement is small. Because affine-invariant Hessian regions [38] are used in [63], each hypothesis of even 5 DoF affine transformation can be generated from only a single pair of corresponding features, which greatly reduces the computational cost of GV.

The above method utilizes local geometry represented by an affine covariant ellipse in GV. Because storing the parameters of ellipse regions significantly increases memory requirement, a method is proposed to learn discretized local geometry representation by minimizing average reprojection error in the space of ellipses in

[95]. It is shown that the representation requires only 24 bits per feature without drop in performance.

Weak Geometric Consistency

Geometric verification explained above is very effective but costly. Therefore, it is only applicable up to a few hundred images. To overcome this problem, Weak Geometric Consistency (WGC) method is proposed in [64, 77], where WGC filters matching features that are not consistent in terms of angle and scale. This is done by estimating rotation and scaling parameters between a query image and a reference image separately assuming the following transformation:

$$\begin{bmatrix} x_q \\ y_q \end{bmatrix} = s \times \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x_p \\ y_p \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{2.30}$$

where $(x_q, y_q)^\top$ and $(x_p, y_p)^\top$ are the positions in query and reference image, s and θ are scaling and rotation parameters, and $(t_x, t_y)^\top$ is translation. In order to efficiently estimate the scaling and rotation parameters, each of feature matches votes to 1D histograms of angle differences and log-scale differences. The score of the largest bin among these two 1D histograms is used as image similarity. This scoring reduces the scores of the images for which the points are not transformed by consistent angles and scales, while a set of points consistently transformed will accumulate its votes in the same histogram bin, keeping a high score.

In WGC, the log-scale difference histogram always has a peak corresponding to 0 (same scale) because most of feature detectors utilizes image pyramid, resulting that most features are detected with the smallest scale. To solve this problem, in [80], the absolute value of the translation $(t_x, t_y)^\top$ is estimated by voting instead of scaling and rotation parameters. In [96–99], similarly, the translation $(t_x, t_y)^\top$ is estimated using 2D histogram instead of shrinking to 1D histogram of its absolute value. Because the scaling and orientaton parameters in Eq. (2.30) are not considered in [96], the method proposed is not scale and rotation invariant.

In [100], a new measure called Pattern Entropy (PE) is introduced, which measures the coherency of symmetric feature matching across the space of two images similar to WGC. This coherency is captured with two histograms of matching orientations, which are composed of the angles formed by the matching lines and horizontal or vertical axis in the synthesized image where two images are aligned horizontally and vertically. As PE is not scale nor rotation invariant, the improved version of PE,

namely Scale and Rotation invariance PE (SR-PE), is proposed in [101]. In SR-PE, rotation and scaling parameters are estimated similar to WGC. However, in SR-PE, these parameters are estimated using two pairs of matching features because it does not utilize scale and orientation parameters of local features. Therefore, it is not applicable to all reference images.

In [102], three types of scoring methods based on weak geometric information are proposed for re-ranking: location geometric similarity scoring, orientation geometric similarity scoring, scale geometric similarity scoring. The orientation and scale geometric similarity scorings are the same as WGC [64, 77]. The location geometric similarity scoring is calculated by transforming the location information into distance ratios to measure the geometric similarity; each of all prossible combination of two matching pairs votes a score to 1D histogram, where each bin is defined by the log ratio of the distances of two points in a query image and corresponding two points in a reference image. The geometric similarity is defined by the score of the bin with miximum votes. The location geometric similarity scoring cannot be integrated with inverted index and is only applicable to re-ranking beucase it requires all combination of matching features.

In [103] spatial-bag-of-features representation is proposed, which is a generalization of the spatial pyramid [66, 104]. The spatial pyramid is not invariant to scale, rotation, nor translation. Spatial-bag-of-features utilizes a specific calibration method which re-orders the bins of a BoVW histogram in order to deal with the above transformations. In [105–107], contextual information is introduced to the BoVW framework by bundling multiple features [105, 107] or extracting feature vectors in multiple scales [106].

Query Expansion

In the text retrieval literature a standard method for improving performance is query expansion, where a number of the highly ranked documents are integrated into a new query. By doing so, additional information can be added to the original query, resulting better search precision. Because the idea of the BoVW framework is comes from the Bag-of-Words (BoW) in text retrieval, it is also natural to borrow query expansion from text retrieval. In [70], various types of query expansions are introduced and compared in the visual domain. There are some insightful observations found in In [70]. Fistly, simply using the top K results for expansion degrades search precision; false positives in the top K results make the expanded queries less informative. Secondly, averaging geometrically verified results for

expansion significantly improves the results because GV excludes false positives from results to the original query. Furthermore, recursively performing this average query expansion further improves the results. Finally, resolution expansion achieves the best performance, which first clusters geometrically verified results into groups, and then issues multiple expanded queries independently created from these groups.

In [108], three approaches are proposed in order to improve query expansion: automatic tf-idf failure recovery, incremental spatial reranking, and context query expansion. In automatic tf-idf failure recovery, after GV, if inlier ratio is smaller than threshold, noisy VWs called *confuser* is estimated according to likelihood ratio. Then, the original query is updated by removing confuser if it improves inlier ratio. In incremental spatial reranking[4], instead of performimg GV to the top K results using the original query, the original query is incrementally updated at each GV if sufficient number of inliers are found in the GV. In context query expansion, a feature outside the bounding box [5] is added to an expanded query if it is consistently found in multiple geometrically verified results.

In [55], discriminative query expansion is proposed, where a linear SVM is trained using geometrically verified results as positive samples and results with lower scores in voting as negative samples. The results are reranked according to the distances from the boundary of the trained SVM.

Summary

In this section, the BoVW extensions were grouped into seven types of approaches: large vocabulary, multiple assignment, post-filtering, weighted voting, geometric verification, weak geometric consistency, and query expansion. These approaches are complementary to each other and often used together. Table 2.1 summarizes the literatures in which these approaches are proposed. We show which approaches are used in the literatures and summarize the best results on publicly available datasets: the UKB[6], Oxford5k[7], Oxford105k, Paris[8], and the Holidays[9] dataset. Oxford105k consists of Oxford5k and 10k distractor images. There are several observations through this summarization:

[4]Incremental spatial reranking is not a query expansion method, but a variant of GV (spatial reranking). Therefore, it is applicable even if query expansion is not used.

[5]Here, it is assumed a query consists of a query image and a bounding box representing the target object of the search.

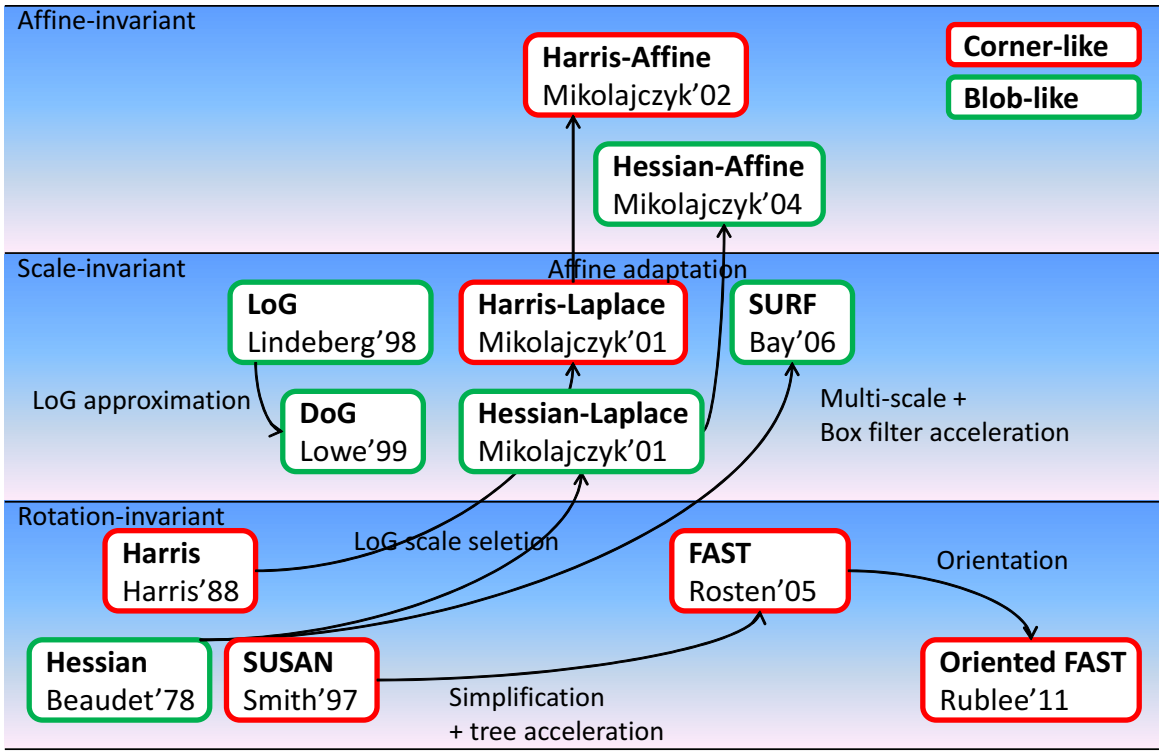
[6]http://vis.uky.edu/ stewe/ukbench/

[7]http://www.robots.ox.ac.uk/ vgg/data/oxbuildings/

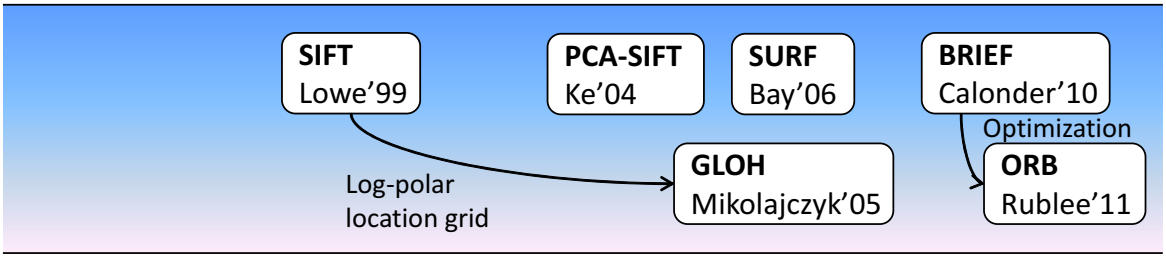
[8]http://www.robots.ox.ac.uk/ vgg/data/parisbuildings/

[9]http://lear.inrialpes.fr/ jegou/data.php

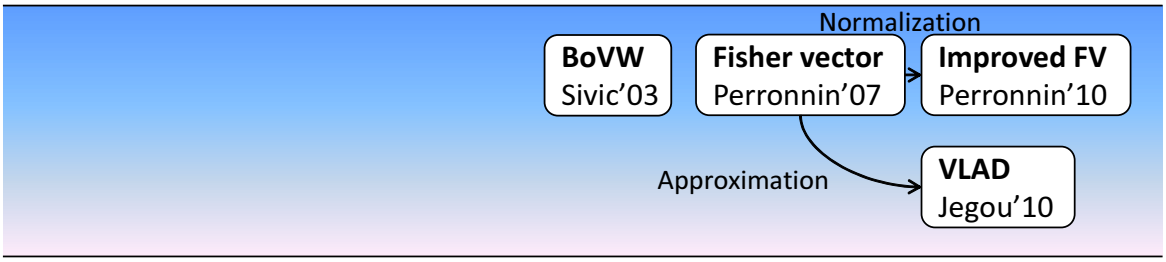
- Large vocabulary or post-filtering is adopted in all of the literatures. These approaches enhance the discreminative power of the BoVW framework and thus are essential for accurate image retrieval system.
- Multiple assignment is also used in many literatures. This is because multiple assignment can improve the recall of feature-level matching at the cost of small increase of computational cost.
- Geometric verification and query expansion are used in many literatures to boost the performance though geometric verification and query expansion are not the main proposal of these literatures. This is because geometric verification and query expansion are needed to achieve the state-of-the-art results on the publicly available datasets. Therefore, we think the absolute values of the accuracies are not directly reflecting the importances of the proposals.
- In several literatures, visual words are learnt using the *test* dataset. Visual words learnt on the test dataset tend to achieve significantly better accuracy than visual words learnt on an independent dataset. When considering the results, we should be aware of this. In Table 2.1, we added '*' mark to the literatures in which visual words are learnt on the datasets.
- While we did not specify the use of RootSIFT [55], RootSIFT has become a de-facto standard descriptor due to its effectiveness and simplicity.



(a) Feature detectors.



(b) Feature descriptors.



(c) Image representations.

Figure 2.1: The overview of the history of local features and image representations. Only an especially important part of local features and image representations is shown.

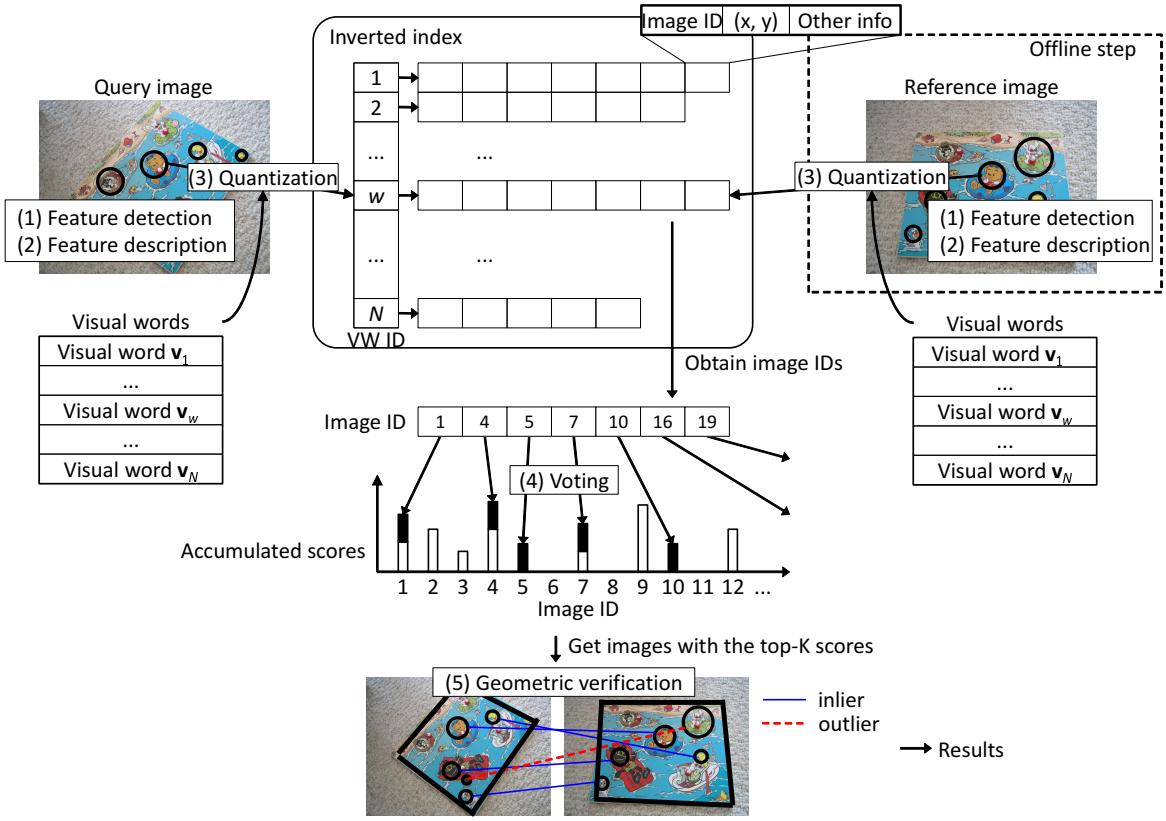
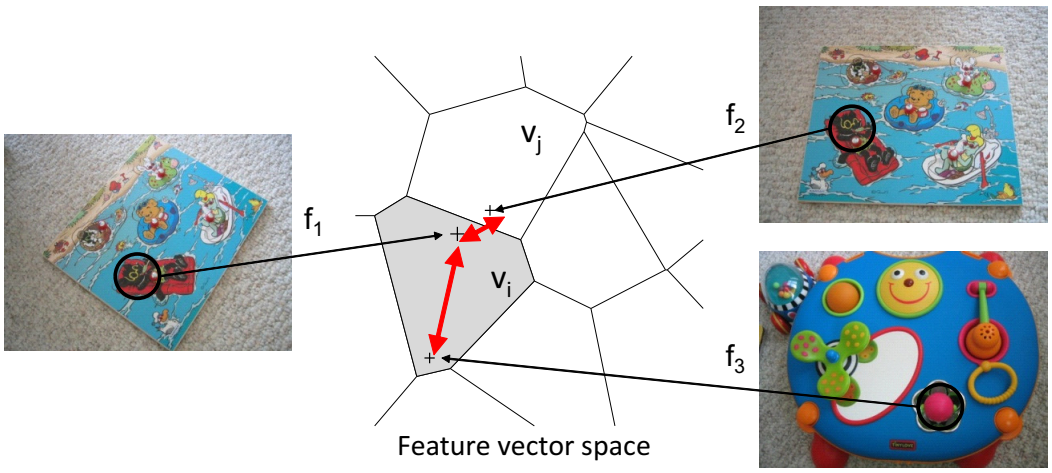
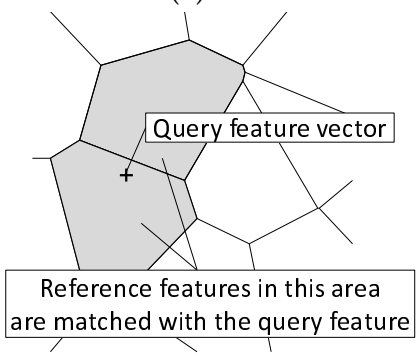


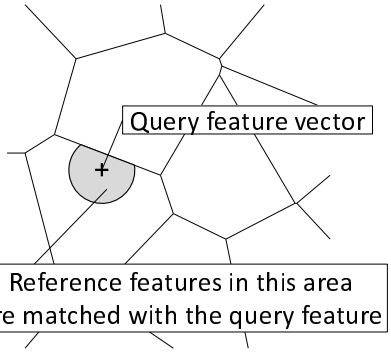
Figure 2.2: A framework of image retrieval using the inverted index data structure.



(a) Problems in the BoVW framework.



(b) Improving matching accuracy by multiple assignment.



(c) Improving matching accuracy by filtering approach.

Figure 2.3: Problems in the BoVW framework and its solutions.

Table 2.1: Summary of existing literature. The use of seven types of approaches is specified: large vocabulary (LV), multiple assignment (MA), post-filtering (PF), weighted voting(WV), geometric verification (GV), weak geometric consistency (WGC), and query expansion (QE). For results, mean average precision (MAP) is shown for each literature and dataset as a performance measurement (higher is better). In addition to MAP, top-4 recall score is also shown for the UKB dataset (higher is better).

		Methods							Results					
Literature	year	LV	MA	PF	WV	GV	WGC	QE	UKB	UKB	Ox5K	Ox105K	Paris6K	Holiday
[62]	2006	✓							3.29					
[63]	2007	✓	✓			✓					0.645			
[79]*	2008	✓	✓			✓		✓			0.825	0.719		
[91]	2009		✓	✓	✓	✓	✓		3.64	0.930	0.685			0.848
[95]*	2009	✓	✓			✓		✓			0.916	0.885		0.780
[77]	2010		✓	✓	✓		✓		3.38	0.870	0.605			0.813
[75]	2010	✓	✓			✓		✓			0.849	0.795	0.824	0.758
[96]	2011	✓				✓	✓				0.713			
[108]	2011	✓				✓		✓			0.827		0.805	
[109]	2011	✓						✓			0.814	0.767	0.803	
[55]*	2012	✓	✓			✓		✓			0.929	0.891	0.910	
[97]*	2012	✓				✓	✓	✓	3.56		0.884	0.864	0.911	
[87]	2013	✓			✓						0.696		0.562	
[84]	2013		✓	✓	✓			✓			0.850	0.816	0.855	0.801
[90]	2013		✓	✓	✓				3.55	0.910				
[110]	2013		✓	✓	✓	✓		✓			0.804	0.750	0.770	0.810
[111]	2013	✓	✓		✓				3.60		0.687			0.809
[112]	2014		✓	✓					3.71					0.840
[99]*	2015	✓				✓	✓	✓			0.950	0.932	0.915	

2.2.2 Fisher Kernel and Fisher Vector

Definition

Fisher kernel is a powerful tool for combining the benefits of generative and discriminative approaches [113]. Let X denote a data item (e.g. a feture vector or a set of feature vector). Here, the generation process of X is modeled by a probability density function $p(X|\lambda)$ whose parameters are denoted by λ. In [113], it is proposed to describe X by the gradient G_λ^X of the log-likelihood function, which is also referred to as the Fisher score:

$$G_\lambda^X = \nabla_\lambda \mathcal{L}(X|\lambda), \tag{2.31}$$

where $\mathcal{L}(X|\lambda)$ denotes the log-likelihood function:

$$\mathcal{L}(X|\lambda) = \log p(X|\lambda). \tag{2.32}$$

The gradient vector describes the direction in which parameters should be modified to best fit the data [114]. A natural kernel on these gradients is the Fisher kernel [113], which is based on the idea of natural gradient [115]:

$$K(X, Y) = G_\lambda^X F_\lambda^{-1} G_\lambda^Y. \tag{2.33}$$

F_λ is the Fisher information matrix of $p(X|\lambda)$ defined as

$$F_\lambda = \mathrm{E}_X[\nabla_\lambda \mathcal{L}(X|\lambda)\ \nabla_\lambda \mathcal{L}(X|\lambda)^\mathrm{T}]. \tag{2.34}$$

Because F_λ^{-1} is positive semidefinite and symmetric, it has a Cholesky decomposition $F_\lambda^{-1} = L_\lambda^\mathrm{T} L_\lambda$. Therefore the Fisher kernel is rewritten as a dot-product between normalized gradient vectors $\mathcal{G}_\lambda^X$ with:

$$\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X. \tag{2.35}$$

The normalized gradient vector $\mathcal{G}_\lambda^X$ is referred to as the Fisher vector of X [116].

GMM Fisher Vector

In [114], the generation process of feature vectors (SIFT) are modeled by the GMM, and the diagonal closed-form approximation of the Fisher vector is derived. Then, the performance of the Fisher vector is significantly improved in [116] by using power-normalization and ℓ_2 normalization. The Fisher vector framework has

achieved promising results and is becoming the new standard in both image classification [116, 117] and image retrieval tasks [118–120].

Let $X = \{x_1, \cdots, x_t, \cdots, x_T\}$ denote the set of low-level feature vectors extracted from an image. and $\lambda = \{w_i, \mu_i, \Sigma_i, i = 1..N\}$ denote the set of parameters for GMM with N components. From Eq. (2.32) and an independence assumption where $x_1, \cdots, x_T$ are independently generated, wehave:

$$\mathcal{L}(X|\lambda) = \sum_{t=1}^{T} \log p(x_t|\lambda). \tag{2.36}$$

The probability that x_t is generated by GMM is:

$$p(x_t|\lambda) = \sum_{i=1}^{N} w_i p_i(x_t|\lambda). \tag{2.37}$$

The i-th component p_i is given by

$$p_i(x_t|\lambda) = \frac{\exp\left(-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right)}{(2\pi)^{D/2}|\Sigma_i|^{1/2}}, \tag{2.38}$$

where D is the dimensionality of the feature vector x_t and $|\cdot|$ denotes the determinant operator. In [114], it is assumed that the covariance matrices are diagonal because any distribution can be approximated with an arbitrary precision by a weighted sum of Gaussians with diagonal covariances.

Let $\gamma_t(i)$ denote the occupancy probability (or posterior probability) of x_t being generated by the i-th component of GMM:

$$\gamma_t(i) = p(i|x_t) = \frac{w_i p_i(x_t|\lambda)}{\sum_{j=1}^{N} w_j p_j(x_t|\lambda)}. \tag{2.39}$$

Letting the subscript d denote the d-th dimension of a vector, Fisher scores corresponding to GMM parameters are obtained as

$$\frac{\partial \mathcal{L}(X|\lambda)}{\partial w_i} = \sum_{t=1}^{T} \left[\frac{\gamma_t(i)}{w_i} - \frac{\gamma_t(1)}{w_1} \right] \text{ for } i \geq 2, \tag{2.40}$$

$$\frac{\partial \mathcal{L}(X|\lambda)}{\partial \mu_{id}} = \sum_{t=1}^{T} \gamma_t(i) \left[\frac{x_{td} - \mu_{id}}{\sigma_{id}^2} \right], \tag{2.41}$$

$$\frac{\partial \mathcal{L}(X|\lambda)}{\partial \sigma_{id}} = \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_{td} - \mu_{id})^2}{\sigma_{id}^3} - \frac{1}{\sigma_{id}} \right]. \tag{2.42}$$

The gradient vector G_λ^X in Eq. (2.31) is obtained by concatenating these partial derivatives.

Next, the normalization terms, the Fisher information matrix F_λ in Eq. (2.34) should be computed. Let f_{w_i}, $f_{\mu_{id}}$, and $f_{\sigma_{id}}$ denote the terms on the diagonal of F_λ which correspond to $\mathcal{L}(X|\lambda)/\partial w_i$, $\partial \mathcal{L}(X|\lambda)/\partial \mu_{id}$, and $\partial \mathcal{L}(X|\lambda)/\partial \sigma_{id}$ respectively. In [114], these terms are obtained approximately as

$$f_{w_i} = T \left(\frac{1}{w_i} + \frac{1}{w_1} \right), \tag{2.43}$$

$$f_{\mu_{id}} = \frac{T w_i}{\sigma_{id}^2}, \tag{2.44}$$

$$f_{\sigma_{id}} = \frac{2 T w_i}{\sigma_{id}^2}. \tag{2.45}$$

The gradient vector in Eq. (2.40) is related to the BoVW because the BoVW can be considered as the relative numbers of occurrences of words given by $\frac{1}{T} \sum_{t=1}^{T} \gamma_t(i)$ ($1 \leq i \leq N$). While the BoVW captures 0-th order statistics, the Fisher kernel also captures 1-st and 2nd order statistics, resulting $(2D+1)N - 1$ dimensional vector. The gradient vector corresponding to 0-th order statistics ($\mathcal{L}(X|\lambda)/\partial w_i$) is sometimes not used because it does not contribute to performance [114]. In this case, the dimensionality of the GMM Fisher vector becomes $2ND$.

Improved Fisher Vector

Although the above Fisher vector has achieved moderate performance, the advantage of this approach is considered to be its efficiency: it can create disctiminative high dimensional vector with small vocabularies (codebook size). However, after

improved Fisher vector is proposed in [116], it becomes widely used in both image classification [117] and image retrieval problems [118, 120]. Improved Fisher vector is calculated by applying two normalizations: power-normalization and ℓ_2 normalization. Power-normalization is to apply the following function to each of dimensions of the original Fisher vector:

$$f(z) = \text{sign}(z)|z|^{\alpha}. \tag{2.46}$$

The value of $\alpha = 0.5$ is often used for reasonable improvement.

Other Extensions

Fisher vectors of the other probabilistic model is also proposed. In [121], the Fisher vectors of Laplacian Mixture Model (LMM) and a Hybrid Gaussian-Laplacian Mixture Model (HGLMM) are proposed. In [122], the Fisher vector is generalized to mixtures of non-Gaussian model in a unified manner. In [123], the Fisher vector of Bernoulli Mixture Model (BMM) is derived. The Fisher vector is also improved by being combined with recent deep learning architechtures in image classification problems [124, 125] and image retrieval problems [126, 127].

2.2.3 Vector of Locally Aggregated Descriptors

In [119], Jégou et al. have proposed an efficient way of aggregating local features into a vector of fixed dimension, namely Vector of Locally Aggregated Descriptors (VLAD). In the construction of VLAD, VWs $c_1, \cdots, c_i, \cdots, c_N$ are first created by the k-means algorithm in the same way as in the BoVW framework. Then, each feature vector x is assigned to the closest VW c_i ($i = \text{NN}(x)$) in the visual codebook, where $\text{NN}(x)$ denotes the identifier of VW closest to x. For each of the visual words, the residual $x - c_i$ from assigned feature vector x is accumulated, and the sums of residuals are concatenated into a single vector, VLAD. More precisely, the VLAD vector v is defined as

$$v_{ij} = \sum_{x \text{ s.t. } \text{NN}(x)=i} \Big[x_j - c_{ij} \Big], \tag{2.47}$$

where x_j and c_{ij} denote the j-th component of the feature vector x and the i-th VW, respectively. Finally, the VLAD vector is ℓ_2-normalized as $v := v/\|v\|_2$. VLAD can be considered as the simplified non-probabilistic version of the partial GMM Fisher vector corresponding to only the parameter μ_{id} [120]. Although the performance of

VLAD is about the same or a little worse than the Fisher vector [120], the VLAD has been widely used in image retrieval due to its simplicity. There is many literature which extends the original VLAD [119]. In the following, these extensions are briefly reviewed.

Modified Normalizations

Many literature focuses on the normalization step in order to improve the VLAD representation. In [120], power-normalization is introduced as for the Fisher vector:

$$v_{ij} := \text{sign}(v_{ij})|v_{ij}|^{\alpha}, \tag{2.48}$$

with $0 \leq \alpha \leq 1$. This power-normalization is followed by ℓ_2 normalization. It have been shown that power-normalization consistently improves the quality of the VLAD representation [120]. One interpretation of this improvement is that it reduces the negative influence of bursty visual elements [91]. Regarding the parameter α, $\alpha = 0.5$ is often used because it empirically shown to lead to near-optimal results. Therefore, power-normalization is also referred to as Signed Square Root (SSR) normalization [128, 129].

In [129], the other normalization is proposed, called intra-normalization. In intra-normalization, the sum of residuals is independently ℓ_2 normalized within each VLAD block v_i:

$$v_{ij} := v_{ij}/\|v_i\|_2. \tag{2.49}$$

Intra-normalization is also followed by ℓ_2 normalization. It is claimed that this normalization completely suppresses the burstiness effect regardless of the amount of bursty elements while power-normalization only discounts the burstiness effect [129]. After power-normalization, the standard deviations of the VLAD vectors become similar among all dimensions. In other words, all dimensions can equally contribute to image similarity, improving the performance of the VLAD representation.

In [130], residual-normalization is proposed, where the residuals are normalized before summation so that all feature vectors contribute equally. With residual-normalization, Eq. (2.47) is modified to

$$v_{ij} = \sum_{x \text{ s.t. } \text{NN}(x)=i} \left[\frac{x_j - c_{ij}}{\|x - c_i\|_2} \right]. \tag{2.50}$$

It is shown that residual-normalization improves the performance of the VLAD representation if it is used in conjunction with power-normalization while it does *not* without power-normalization [130]. In [131], triangulation embedding is proposed. It can be seen a modified version of VLAD, where normalized residuals from *all* centroids are aggregated as

$$v_{ij} = \sum_{x} \left[\frac{x_j - c_{ij}}{||x - c_i||_2} \right]. \tag{2.51}$$

It is similar to residual-normalization in Eq. (2.50) but only residuals from the nearest centroids are aggregared in Eq. (2.50).

Other Extentions

In [132], it is proposed to modify the summation term in Eq. (2.47) to mean or median operations. It is claimed that the mean aggregation outperforms the original sum aggregation in terms of an image-level Receiver Operating Characteristic (ROC) curve analysis. However, in [133], it is shown that the original sum aggregation is still better in terms of image retrieval performance.

In [128, 120], it is proposed to perform Principal Component Analysis (PCA) to input vector x before aggregation, which decorrelates and whiten input vector. Decorrelating the input vector x is very important because the VLAD implicitly assumes that the covariance matrices of x is isotropic. In [130], Local Coordinate System (LCS) is proposed, where the residuals to be summed are rotated by a rotation matrix Q_i

$$v_{ij} = \sum_{x \text{ s.t. } \mathrm{NN}(x)=i} Q_i \left[x_j - c_{ij} \right]. \tag{2.52}$$

The VW-specific rotation matrix Q_i is obtained by learning a local PCA per VW. This is a contrast to the approach in [120], where the input vector x is rotated by a globally learnt PCA matrix. LCS has no effect if power-normalization is not applied ($\alpha = 1$). However, it is shown that LCS with power-normalization outperforms the global rotation with power-normalization.

2.3 Deep Learning for Image Retrieval

Starting from ImageNet Large-scale Visual Recognition Challenge (ILSVRC) in 2012[10], where Convolutional Neural Networks (CNN) [134] had beaten the traditional state-of-the-art framework (i.e. SIFT feature + Fisher vector), CNN has become a de facto standard for image recognition tasks. Following this trend, the deep learning approach also began to be applied to image retrieval tasks. In this section, we briefly review recent deep learning approaches related to image retrieval.

Early works that have applied deep learning to image retrieval can be found in [135–138, 126]. In [135], the CNN architecture used in [134] is applied for image retrieval. Different from image recognition tasks, the best performance is achieved at the layer that is two levels below the outputs, not the very top of the network, which is consistent with the results of subsequent papers[11]. In [135], the performance of CNN in [134] is reported to be comparable to the Fisher vector or VLAD methods so far. In [136–138], a comparative study of CNN and Fisher vector is performed. In [126], manys best practices for CNN is presented.

While the above methods utilizes CNN to extract a single global feature, there are different approaches to achieve better performance. In [139], it is shown that CNN can be applied to keypoint prediction task and find correspondences between objects. In [140, 141], CNN features are densely extracted and the Fisher vector or VLAD framework is used for pooling (aggregation). In [142], the dense CNN features from multiple networks are indexed by traditional inverted index. In [143], a unified framework for both of image retrieval and classification is proposed, where CNN features are extracted from multiple object proposals for each image, and the Naive-Bayes Nearest-Neighbor (NBNN) search [144] is performed to calculate the distance between a query image and reference images. In [145], *convolutional features* are extracted from every position of different layers on CNN, and then these features are encoded by the VLAD framework. It is reported that this framework achieves the best performance in very low-dimensional representation. While the above methods utilizes Fisher vector or VLAD for aggregation, in [146], it is claimed that the simple aggregation method based on sum pooling provides the best performance for deep convolutional features.

Deep learning is also applied for patch-level tasks. In [147], in order to extract patch-level descriptors, Mairal et al. proposed a deep convolutional architecture

[10] http://www.image-net.org/challenges/LSVRC/2012/

[11] In many papers, it is reported that the best performance has been achieved at the first fully connected layer.

based on Convolutional Kernel Network (CKN) [148], which is an unsupervised framework to learn convolutional architectures. In [149], a Siamese network is used to learn discriminant patch representations, where an aggressive mining strategy is adopted to handle *hard* negative and *hard* positive pairs. In [150], similarity between image patches is directly learnt with CNN. While several types of network are proposed and compared, it is reported that a two-channel and two-stream architecture achieved the best performance, where two patches to be compared are fed to the first convolutional layer directly (cf. a Siamese network). For each patch, two regions with different scales are used as an input of the network. Similarly, in [151], a patch matching system called MatchNet is proposed to learn a CNN for local feature description as well as a network for robust feature comparison. In [152], a new regression-based approach is proposed to extract feature points that are especially robust repeatable under temporal changes. In [153], a learning scheme based on CNN is introduced to estimate a canonical orientation for local features. Because it is difficult to explicitly define a *correct* canonical orientation, it is proposed to implicitly define a canonical orientation to be learnt such that minimizes the distances between descriptors of correct feature pairs. In [154], a DNN architecture is proposed that combines the three components of standard pipelines for local feature matching, i.e detection, orientation assignment, and description, into a single differentiable network.

There are several approaches to compress CNNs in order to reduce memory requirements and/or speed up the recognition. In [155], it is proposed to utilize product quantization [82] to compress CNN. In [156], a low-rank matrix approximation is used to compress CNNs. In [157], it is proposed to binarize CNNs and input signals to compress CNNs and achieve faster convolutional operations. In [158, 159], pruning, quantization, and huffman coding is applied to CNNs to achieve an energy-efficient engine.

Chapter 3

Fisher Vector for Binary Features

Recently, the Fisher vector representation of local features has attracted much attention because of its effectiveness in both image classification and image retrieval. Another trend in the area of image retrieval is the use of binary features, such as ORB, FREAK, and BRISK. Considering the significant performance improvement for accuracy in both image classification and retrieval by the Fisher vector of continuous feature descriptors, if the Fisher vector were also to be applied to binary features, we would receive similar benefits in binary feature-based image retrieval and classification. In this chapter, we derive the closed-form approximation of the Fisher vector of binary features modeled by the Bernoulli mixture model. We also propose accelerating the Fisher vector by using the approximate value of posterior probability. Experiments show that the Fisher vector representation significantly improves the accuracy of image retrieval compared with a bag of binary words approach.

3.1 Introduction

With the advancement of both stable interest region detectors [27] and robust and distinctive descriptors [26], local feature-based image or object retrieval has attracted a great deal of attention. In local feature-based image retrieval or recognition, each image is first represented by a set of local features $X = \{x_1, \cdots, x_t, \cdots, x_T\}$, where T is the number of local features. The set of features X is then encoded into a fixed length vector in order to calculate any (dis)similarity between sets of features. The most frequently used method is a bag-of-visual words (BoVW) representation [61], where feature vectors are quantized into visual words (VWs) using a visual codebook that result in a histogram representation of VWs.

Recently, the Fisher vector representation [114] has attracted much attention because of its effectiveness. The Fisher vector is defined by the gradient of log-likelihood function normalized with the Fisher information matrix. In [114], feature vectors are modeled by the Gaussian mixture model (GMM), and a closed form approximation is first proposed for the Fisher information matrix of GMM. Then, the performance of the Fisher vector is improved in [116] by using power and ℓ_2 normalization. Because the Fisher vector can represent higher order information than the BoVW representation, it has been shown that it can outperform the BoVW representation in both image classification [116] and image retrieval tasks [118–120].

Another trend in the area of image retrieval is the use of binary features, such as Oriented FAST and Rotated BRIEF (ORB) [19], Fast Retina Keypoint (FREAK) [21], Binary Robust Invariant Scalable Keypoints (BRISK) [20], KAZE features [160], Accelerated-KAZE (A-KAZE) [22], Local Difference Binary (LDB) [161, 162], and Learned Arrangements of Three patCH codes (LATCH) [23]. In addition to these methods that extract binary features directly, there are many methods that encode continuous feature vectors (e.g., SIFT) into compact binary codes [163–169]. Binary features are one or two orders of magnitude faster than the Scale Invariant Feature Transform (SIFT) [44] or Speeded Up Robust Features (SURF) [47] features in detection and description, while providing comparable performance [19, 30]. These binary features are especially suitable for mobile visual search or augmented reality on mobile devices [161]. While the Fisher vector is widely applied to continuous features (e.g., SIFT) that can be modeled by GMM, to the best of our knowledge, there has been no attempt to apply the Fisher vector to the abovementioned recent binary features for the purpose of image retrieval. Considering the significant performance improvement for accuracy in both image classification and retrieval by the Fisher vector of continuous features, if the Fisher vector were also to be applied to binary features, we would receive similar benefits in binary feature-based image retrieval and classification.

In this chapter, we propose to apply the Fisher vector representation to binary features to improve the accuracy of binary feature-based image retrieval. Table 3.1 shows the position of the proposal of this chapter. Our main contribution is to model binary features using the Bernoulli mixture model (BMM) and derive the closed-form approximation of the Fisher vector of BMM [170]. Experimental results show that the proposed Fisher vector outperforms the BoVW method on various types of objects. In addition, we also propose a fast approximation method to accelerate the computation of the proposed Fisher vectors by one order of magnitude with

Table 3.1: Position of this chapter.

Feature type	BoVW	Fisher Vector
Continuous	[61]	[114]
Binary	[17]	This chapter

comparable performance. In the experiments, we evaluate the effectiveness of both the proposed Fisher vector representation of binary features and their associated vector normalization method. In particular, we demonstrate that a normalization method, originally proposed for the other vector representation, also works well for the proposed Fisher vector. The proposed Fisher vector representation of binary features is general and not restricted to image features; it is also expected to be applicable to other modalities such as audio signals [171, 172].

3.2 Fisher Vector for Binary Features

In this section, we model binary features with the Bernoulli distribution, and derive the Fisher vector representation of binary features.

3.2.1 Bernoulli Mixture Model

Let x_t denote a D-dimensional binary feature out of T binary features $X = \{x_1, \cdots, x_t, \cdots, x_T\}$ extracted from an image. In modeling binary features, it is straightforward to adopt a single multivariate Bernoulli distribution. However, although many binary descriptors are designed so that bits of resulting binary features are uncorrelated [19], there are still strong dependencies among bits. Therefore, a single multivariate Bernoulli component will be inadequate to cope with the kind of complex bit dependencies that often underlie binary features. This drawback is overcome when several Bernoulli components are adequately mixed. In this chapter, we propose to model binary features with Bernoulli mixture model (BMM). The use of BMM instead of a single multivariate Bernoulli distribution will be justified in the experimental section.

Let $\lambda = \{w_i, \mu_{id}, i = 1, \cdots, N, d = 1, \cdots, D\}$ denote a set of parameters for a multivariate Bernoulli mixture model with N components, and x_{td} represents the d-th bit of x_t. Given the parameter set λ, the probability density function of T binary

features X is described as:

$$
\begin{aligned}
p(X|\lambda) &= \prod_{t=1}^{T} p(x_t|\lambda), \\
p(x_t|\lambda) &= \sum_{i=1}^{N} w_i p_i(x_t|\lambda), \\
p_i(x_t|\lambda) &= \prod_{d=1}^{D} \mu_{id}^{x_{td}} (1-\mu_{id})^{1-x_{td}}.
\end{aligned} \tag{3.1}
$$

In order to estimate the values of the parameter set λ, given a set of training binary features $x_1, \cdots, x_s, \cdots, x_S$, the expectation-maximization (EM) algorithm is applied [173]. In the expectation step, the occupancy probability $\gamma_s(i)$ (or posterior probability $p(i|x_s, \lambda)$) of x_s being generated by the i-th component of BMM is calculated as

$$
\gamma_s(i) = p(i|x_s, \lambda) = \frac{w_i p_i(x_s|\lambda)}{\sum_{j=1}^{N} w_j p_j(x_s|\lambda)}. \tag{3.2}
$$

In the maximization step, the parameters are updated as

$$
S_i = \sum_{s=1}^{S} \gamma_s(i), \;\; w_i = S_i/S, \;\; \mu_{id} = \frac{1}{S_i} \sum_{s=1}^{S} \gamma_s(i) x_{sd}. \tag{3.3}
$$

In our implementation, parameter w_i is initialized with $1/N$, and μ_{id} is with uniform distribution $U(0.25, 0.75)$. From our experience, these initial parameters do not have a large impact on the final result.

3.2.2 Deriving the Fisher Vector of BMM

In this section, we derive the Fisher vector of BMM. In order to calculate the Fisher vector $\mathcal{G}_\lambda^X$ in Eq. (2.35), the Fisher score G_λ^X in Eq. (2.31) and the Fisher information matrix F_λ in Eq. (2.34) should be calculated.

Letting $G^X_{\mu_{id}}$ denote the Fisher score w.r.t. the parameter $\mu_{id} \in \lambda$, $G^X_{\mu_{id}}$ is calculated as:

$$G^X_{\mu_{id}} = \frac{1}{T}\frac{\partial \mathcal{L}(X|\lambda)}{\partial \mu_{id}} \tag{3.4}$$

$$= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial \mathcal{L}(x_t|\lambda)}{\partial \mu_{id}}$$

$$= \frac{1}{T}\sum_{t=1}^{T}\frac{1}{p_i(x_t|\lambda)}\frac{\partial p_i(x_t|\lambda)}{\partial \mu_{id}}, \tag{3.5}$$

where

$$\frac{\partial p_i(x_t|\lambda)}{\partial \mu_{id}} = (-1)^{1-x_{td}}\prod_{e=1,e\neq d}^{D}\mu_{ie}^{x_{te}}(1-\mu_{ie})^{1-x_{te}}. \tag{3.6}$$

Finally we obtain:

$$G^X_{\mu_{id}} = \frac{1}{T}\sum_{t=1}^{T}\gamma_t(i)\frac{(-1)^{1-x_{td}}}{\mu_{id}^{x_{td}}(1-\mu_{id})^{1-x_{td}}}, \tag{3.7}$$

where $\gamma_t(i)$ is the occupancy probability defined in Eq. (3.2).

Then, we derive the approximate Fisher information matrix of BMM under the following three assumptions [114]: (1) the Fisher information matrix F_λ is diagonal, (2) the number of binary features x_t extracted from an image is constant and equal to T, and (3) the occupancy probability $\gamma_t(i)$ is peaky; there is one index i such that $\gamma_t(i) \approx 1$ and that $\forall j \neq i$, $\gamma_t(j) \approx 0$.

As we assume the Fisher information matrix is diagonal, Eq. (2.34) is approximated as $F_\lambda \approx \mathrm{diag}(F_{\mu_{11}}, \cdots, F_{\mu_{ND}})$, where $F_{\mu_{id}}$ denotes the Fisher information w.r.t. μ_{id}:

$$F_{\mu_{id}} = \mathrm{E}\left[\left(\frac{\partial \mathcal{L}(X|\lambda)}{\partial \mu_{id}}\right)^2\right]. \tag{3.8}$$

Assuming that binary features are independently generated and the number of binary features is T, Eq. (3.8) can be expanded as:

$$F_{\mu_{id}} = \mathrm{E}\left[\left(\frac{\partial \mathcal{L}(X|\lambda)}{\partial \mu_{id}}\right)^2\right] \tag{3.9}$$

$$= \mathrm{E}\left[\left(\sum_{t=1}^{T} \frac{\partial \mathcal{L}(x_t|\lambda)}{\partial \mu_{id}}\right)^2\right] \tag{3.10}$$

$$= \sum_{t=1}^{T} \mathrm{E}\left[\left(\frac{\partial \mathcal{L}(x_t|\lambda)}{\partial \mu_{id}}\right)^2\right] + 2 \sum_{1 \le t < s \le T} \mathrm{E}\left[\frac{\partial \mathcal{L}(x_t|\lambda)}{\partial \mu_{id}}\right] \mathrm{E}\left[\frac{\partial \mathcal{L}(x_s|\lambda)}{\partial \mu_{id}}\right]. \tag{3.11}$$

As the parameter set λ is estimated with maximum-likelihood estimation, we have:

$$\mathrm{E}\left[\frac{\partial \mathcal{L}(x_t|\lambda)}{\partial \mu_{id}}\right] = 0.$$

Using the value of the Fisher score in Eq. (3.7), we get:

$$\mathrm{E}\left[\left(\frac{\partial \mathcal{L}(x_t|\lambda)}{\partial \mu_{id}}\right)^2\right] = \int_{x_t} p(x_t|\lambda) \frac{\gamma_t^2(i)}{\left(\mu_{id}^{x_{td}}(1-\mu_{id})^{1-x_{td}}\right)^2} dx_t \tag{3.12}$$

$$= \int_{x_{td}=1} p(x_t|\lambda) \frac{\gamma_t^2(i)}{\mu_{id}^2} dx_t + \int_{x_{td}=0} p(x_t|\lambda) \frac{\gamma_t^2(i)}{(1-\mu_{id})^2} dx_t. \tag{3.13}$$

Using the assumption that the occupancy probability $\gamma_t(i)$ is peaky, we approximate $\gamma_t^2(i)$ as $\gamma_t(i)$. Using the following equations,

$$\int_{x_{td}=1} p(x_t|\lambda)\gamma_t(i) dx_t = w_i \sum_{j=1}^{N} w_j \mu_{jd}, \tag{3.14}$$

$$\int_{x_{td}=0} p(x_t|\lambda)\gamma_t(i) dx_t = w_i \sum_{j=1}^{N} w_j (1-\mu_{jd}), \tag{3.15}$$

we finally obtain:

$$F_{\mu_{id}} = T w_i \left(\frac{\sum_{j=1}^{N} w_j \mu_{jd}}{\mu_{id}^2} + \frac{\sum_{j=1}^{N} w_j (1-\mu_{jd})}{(1-\mu_{id})^2} \right).$$

The Fisher vector $\mathcal{G}_\lambda^X$ is obtained with the concatenation of normalized Fisher scores $F_{\mu_{id}}^{-1/2} G_{\mu_{id}}^X$ $(i = 1, \cdots, N, d = 1, \cdots, D)$.

3.2.3 Vector Normalization

The Fisher vector is further normalized with power normalization and ℓ_2 normalization [116]. Given a Fisher vector $z = \mathcal{G}_\lambda^X$, the power-normalized vector $f(z)$ is calculated as

$$f(z) = \text{sign}(z)|z|^\alpha. \tag{3.16}$$

In experiments, we set $\alpha = 0.5$ as recommended in [116]. After the power normalization, ℓ_2 normalization is performed to $f(z)$, resulting in the final Fisher vector representation of the set of binary features. In addition, we propose to use intra normalization [129] for this Fisher vector instead of the power and ℓ_2 normalization. The intra normalization method was originally proposed for the VLAD representation described in Section 2.2.3, not for the Fisher vector. However, the purpose of intra normalization is to alleviate the problem of burstiness in visual words [91, 129] and, this is the same as that of power normalization. Therefore, it is also expected to work well for the Fisher vector. In the case of the Fisher vector, intra normalization is done by performing ℓ_2 normalization within each BMM component.

3.2.4 Fast Approximated Fisher Vector

The most computationally expensive part of the proposed Fisher vector is the calculation of the occupancy probability $\gamma_t(i)$ in Eq. (3.7) because $F_{\mu_{id}}$ does not depend on the input vector X and can be precomputed. In this chapter, we also propose to accelerate the proposed Fisher vector by using the approximate value of $\gamma_t(i)$.

Firstly, each i-th component of BMM is converted into a representative binary vector $y_i = (y_{i1}, \cdots, y_{iD})$ as

$$y_{id} = \begin{cases} 1 & \mu_{id} \geq 0.5 \\ 0 & \mu_{id} < 0.5. \end{cases} \tag{3.17}$$

Then, for each $x_t \in X$, the most similar representative binary vector $y_{\hat{i}}$ is calculated by $\hat{i} = \arg\min_i |x_t - y_i|$. This involves only the calculation of Hamming distance and can be done very fast. Finally, we obtain approximated $\gamma'_t(i)$ as

$$\gamma'_t(i) = \begin{cases} 1 & i = \hat{i} \\ 0 & i \neq \hat{i}. \end{cases} \tag{3.18}$$

This approximation is also based on the assumption that the occupancy probability $\gamma_t(i)$ is peaky.

Figure 3.1: Example images from the Stanford MVS dataset. Images in the top row are examples of reference images and images in the bottom row are examples of query images.

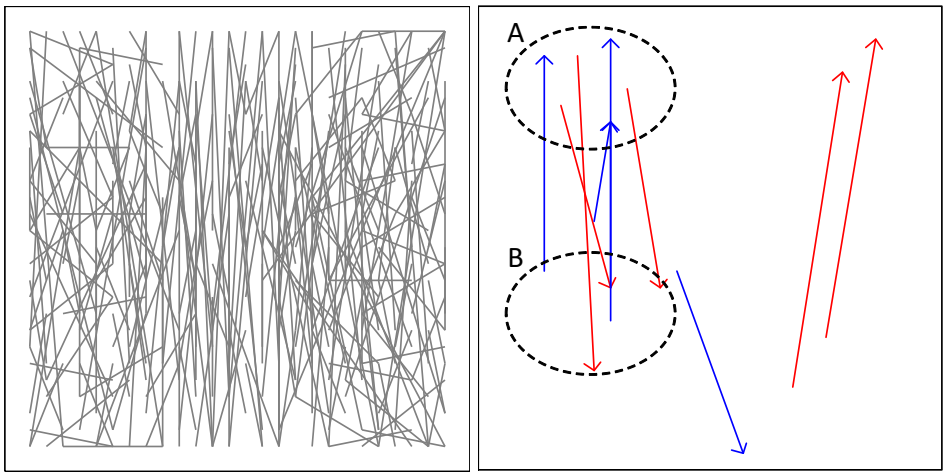
3.3 Experiment

In the experiments, the Stanford mobile visual search dataset[1] is used to evaluate the effectiveness of the proposed Fisher vector in image retrieval. The dataset contains camera-phone images of CD covers, books, business cards, DVD covers, outdoor landmarks, museum paintings, print documents, and video clips. While it includes eight classes of images, we mainly use general CD class images in this chapter. These images consist of 100 reference images and 400 query images. Because some query images are too large (10M pixels), all images are resized so that the longest sides of the images are less than 640 pixels, while keeping the original aspect ratio. Figure 3.1 shows example images from the dataset.

Dissimilarity between two images is defined by the Euclidean distance between either the BoVW or the Fisher vector representations of the images. As an indicator of the retrieval performance, mean average precision (MAP) [62, 77] is used. For each query, a precision-recall curve is obtained based on the retrieval results. Average precision is calculated as the area under the precision-recall curve. Finally, the MAP score is calculated as the mean of the average precisions over all queries.

As a binary feature, we adopt the ORB [19] descriptor, which is one of the most frequently used binary features. An implementation of the ORB descriptor is available in an open source library[2]. On average, 900 features are extracted from

[1] http://www.stanford.edu/~dmchen/mvs.html
[2] http://opencv.org/



(a) (b)

Figure 3.2: (a) All point pairs of 256 binary tests used in the ORB descriptor. (b) Five tests corresponding to the bits with the top five probabilities μ_{id} of being 1 (red) and 0 (blue). A randomly selected component out of $N = 32$ components is shown.

four scales. The parameter set λ is estimated with the EM algorithm using one million ORB binary features extracted from the MIR Flickr collection[3]. The following experiments were performed on a standard desktop PC with a Core i7 970 CPU.

3.3.1 Evaluating BMM in terms of Log-likelohood

First, we evaluate the validity of modeling binary features with BMM. To do this, we use 1M training binary features to estimate the parameter set of BMM and use 100K testing binary features to calculate log-likelihood for five different binary features: BIREF [57], BRISK [20], FREAK [21], ORB [19], and Random. The default parameters in OpenCV 3.0 are used for these binary features. Here, Random is an ideal binary feature that is artificially created so that the mean value of each bit becomes 0.5 and bits are independent. Figure 3.4 shows mean log-likelihood $1/T \log p(X|\lambda)$ of BMM for different binary features in terms of N, where X is the set of testing binary features with the size of T. We can see that the mean log-likelihood for Random is constant at all N and is almost the same as theoretical value $\log 1/2^{128} \approx -69.3$. This indicates that it is not appropriate to model really ideal binary features with BMM because there is no underlying bit dependency. On the contrary, for the other *real* binary features, the mean log-likelihood becomes higher as N increases. This implies that there are complex underlying bit dependencies in these binary features and

[3]http://press.liacs.nl/mirflickr/

BMM with larger N can capture these dependencies more appropriately. Among four real binary features, the mean log-likelihood for BRIEF is larger than that for the others. This result is reasonable because the BRIEF feature is created by random binary tests that are not optimized, resulting in highly dependent bits and thus BMM works more effectively. On the other hand, the mean log-likelihood for ORB is the smallest among the other real binary features. This is because the ORB descriptor is highly optimized so that the average value of each resulting bit is close to 0.5, and bits are not correlated. From this point of view, we can say that the ORB descriptor is the closest to the ideal binary feature. In the following experiments, we focus on the ORB features.

3.3.2 Clustreing Effect

We investigate the clustering results generated from the estimation of the parameter set λ of BMM with $N = 32$ components. Figure 3.2 (a) represents all point pairs (a_d, b_d) of the 256 binary tests used in the ORB descriptor explained in Section 2.1.2. Figure 3.2 (b) visualizes a part of the parameter sets λ of a randomly selected component out of $N = 32$ components. In each figure, red (blue) arrows represent five tests corresponding to the five largest (smallest) μ_{id}. The arrows are drawn from a_d to b_d, and μ_{id} represents the probability that a_d is brighter than b_d. Therefore, the pixel at the head of a red arrow tends to be brighter than the tail of the red arrow while the pixel at the head of a blue arrow tends to be darker than the tail of the blue arrow. We can see that the binary tests with the largest and the smallest μ_{id} concentrate on small areas (e.g., between the areas A and B in Figure 3.2 (b)). Thus, Figure 3.2 implies that some bits of the ORB descriptor are highly correlated and that BMM successfully captures this correlation. The result justifies the use of BMM instead of single multivariate Bernoulli distribution to model binary features. Figure 3.3 visualizes a part of the parameter sets λ corresponding to all of the $N = 32$ components.

3.3.3 Impact of Normalization

The performance of the Fisher vector of binary features is evaluated in terms of image retrieval accuracy. In particular, the effect of the normalization methods described in Section 3.2.3 is investigated. The following six methods are compared: (1) bag of binary words approach with 1024 centroids (BoBW) [17, 18], (2) Fisher vector without normalization (FV), (3) Fisher vector with ℓ_2 normalization (L2 Norm), (4)

Table 3.2: Comparison of the proposed method with the BoVW method on eight classes

	cd	book	card	dvd
Prop	0.785	0.892	0.377	0.895
BoVW	0.623	0.625	0.179	0.472
	landmark	painting	document	video
Prop	0.179	0.676	0.472	0.840
BoVW	0.064	0.482	0.208	0.559

Fisher vector with power normalization (P Norm), and (5) Fisher vector with both power and ℓ_2 normalization (P+L2 Norm). (6) Fisher vector with intra normalization (In Norm).

Figure 3.5 shows a comparison of the Fisher vector and BoBW representations applied to binary features on eight classes. The accuracy of the Fisher vector without any normalization (FV) is disappointing compared with the BoBW framework. If ℓ_2 or power normalization is adopted, the accuracy of the Fisher vector is significantly improved. The combination of the two normalizations further improves the performance, which is consistent with the case of SIFT+GMM [116]. A little surprisingly, in many casese, the intra normalization method outperforms the other normalization methods. With appropriate normalization methods, the accuracy improves as the number N of components increases, which is also consistent with the case of SIFT+GMM [120]. Table 3.2 shows the accuracy of the proposed method (In Norm, $N = 512$) and the BoVW method on eight classes. We can see that the proposed Fisher vector consistently outperforms the BoVW method on different datasets. In particular, the difference of accuracy between the proposed method and the BoVW method is relatively larger for book, card, dvd, document, and video classes. These classes includes many simple edges and corners (e.g., logo or text), and binary features extracted from these edges and corners are similar to each other. In the case of BoVW method, these binary features tend to be quantized into the same VW and less discriminative. On the other hand, the proposed Fisher vector can capture the "difference" from the components of BMMs; therefore it is more discriminative, resulting in better results.

3.3.4 Performance for Various String Lengths

Next, we investigate the impact of the length of the binary strings on accuracy. In this experiment, the first D' bits of the full 256-bit string are used. This is reasonable

because, in the ORB algorithm, binary tests are sorted according to their entropies; the leading bits are more important.

Figure 3.6 shows the accuracy of the Fisher vector with intra normalization ($N = 512$) as a function of the number of components, where the length of binary strings D' varies from 16 to 256. We can see that the Fisher vector of longer binary strings achieves better accuracy. However, the gain becomes smaller as the binary string becomes longer. This is because the ending bits tend to be correlated to other bits and thus are less informative. Therefore, we can use shorter strings for efficiency at the cost of accuracy. Because the computational cost of the Fisher vector is proportional to ND'[4], the other choice to reduce the computational cost is to use smaller N. Figure 3.6 indicates that it is better to use shorter strings down to 64-bit strings instead of using smaller N. For instance, the MAP score of 0.733 at $N = 256$ and $D' = 64$ is better than that of 0.714 at $N = 64$ and $D' = 256$. Otherwise, it is better to use smaller N instead of using smaller D', e.g. 0.639 at $N = 256$ and $D' = 32$ v.s. 0.663 at $N = 32$ and $D' = 256$. However, in this paper, we use full 256-bit strings in the other experiments to make the most of the ORB descriptor.

3.3.5 Increasing Database Size

We investigate the performance of the Fisher vector when the size of database becomes large. In order to increase the size of database, we use images in the MIR Flickr collection as a distractor in the same way as in [77]. Figure 3.7 compares the Fisher vector with intra normalization ($N = 512$, $D' = 256$) and the BoBW representation with the different numbers of distractors. In Figure 3.7, 0, 100, 1,000, and 10,000 distractor images are added to 100 reference images, resulting that the size of database becomes 100, 200, 1,100, and 10,100 respectively. We can see that the Fisher vector achieves better performance for all database sizes. Although the accuracy of both the Fisher vector and the BoBW representations drops as the size of the database increases, the degradation of the Fisher vector is relatively small. This is because the Fisher vector can represent higher order information than the BoBW representation and is more discriminating even with larger database sizes. It can be said that the effectiveness of the proposed Fisher vector representation becomes more significant when the database size is increased.

[4]This is required in calculating the occupancy probability $\gamma_t(i)$, which is the most computationally expensive part of the propsoed Fisher vector as described in Section 3.2.4.

Table 3.3: Average degradation of the approximated Fisher vector on eight classes

cd	book	card	dvd
6.4%	4.9%	6.8%	8.3%
landmark	painting	document	video
3.0%	4.1%	7.8%	3.9%

3.3.6 Evaluation of Fast Approximated Fisher Vector

Finally, we evaluate the performance of the approximated Fisher vector described in Section 3.2.4. Figure 3.8 compares the approximated and exact Fisher vector with intra normalization ($D' = 256$). The distractor images in Section 3.3.5 are not used in this experiment. It can be seen that the approximated Fisher vector is one order of magnitude faster than the exact Fisher vector while the degradation of accuracy is only 6.4% on average and 1.6% for $N = 512$. Table 3.3 shows average degradation of the MAP score on eight classes when the approximated Fisher vector with intra normalization ($D' = 256$) is adopted. We can see that there is not so much difference in degradation of accuracy among different classes.

This approximated Fisher vector uses two approximations. The first one is the approximation of the occupancy probability $\gamma_t(i)$; we approximate $\gamma_t(i)$ with the largest value to 1 and the others to 0 as Eq. (3.18). This is based on the assumption that $\gamma_t(i)$ is peaky. Figure 3.9 (a) shows the distribution of maximum occupancy probability $\max_i \gamma_t(i)$ for $N = 512$. We can confirm that $\max_i \gamma_t(i)$ is near to 1 in most cases as expected. The other approximation is that the component with maximum occupancy probability is approximately obtained as $\hat{i} = \arg\min_i |x_t - y_i|$ using representative binary vectors defined in Eq. (3.17). Figure 3.9 (b) shows the accuracy of this approximation; the probability that $\arg\min_i |x_t - y_i| = \arg\max_i \gamma_t(i)$. We can see that although the accuracy declines as N increases, the accuracy of this approximation is still 57% even for $N = 512$. As the final approximated Fisher vector is created using a number of feature vectors, this approximation works well as shown in Figure 3.8. The final approximated Fisher vector is created using a number of feature vectors, this approximation works well as shown in Figure 3.8.

3.4 Summary

In this chapter, we proposed the application of the Fisher vector representation to binary features to improve the accuracy of binary feature-based image retrieval.

We derived the closed-form approximation of the Fisher vectors of binary features as modeled by the Bernoulli mixture model. In addition, we also proposed a fast approximation method that accelerates the computation of the proposed Fisher vectors by one order of magnitude with comparable performance. The effectiveness of the Fisher vectors of binary features was confirmed. There were some interesting observations; for example, the performance of the Fisher vector without power and ℓ_2 normalization was very poor, while the Fisher vector with power and ℓ_2 normalization outperformed the BoBW framework. The effectiveness of the proposed Fisher vector representation becames more significant when the database size increased. Furthermore, we have demonstrated that the intra normalization method originally proposed for VLAD also worked well for the proposed Fisher vector and outperformed the conventional normalization methods. This result encourages us to apply the intra normalization method to the Fisher vector of GMM. In future, we will apply the Fisher vector of binary features to image classification problems. We also expect that the proposed Fisher vector representation can also be successfully applied to other modalities such as audio signals.

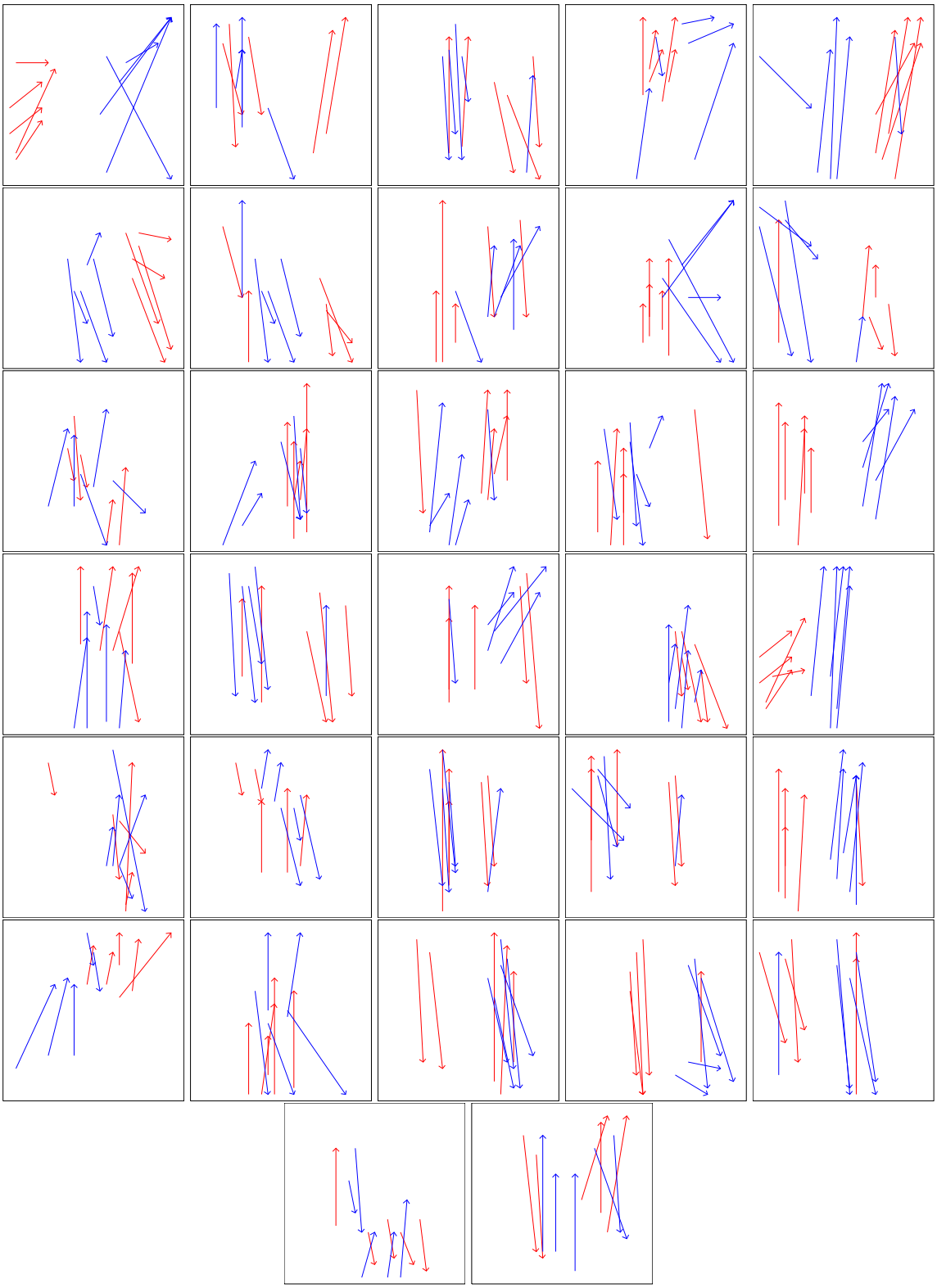


Figure 3.3: Five tests corresponding to the bits with the top five probabilities μ_{id} of being 1 (red) and 0 (blue). $N = 32$ components are shown.

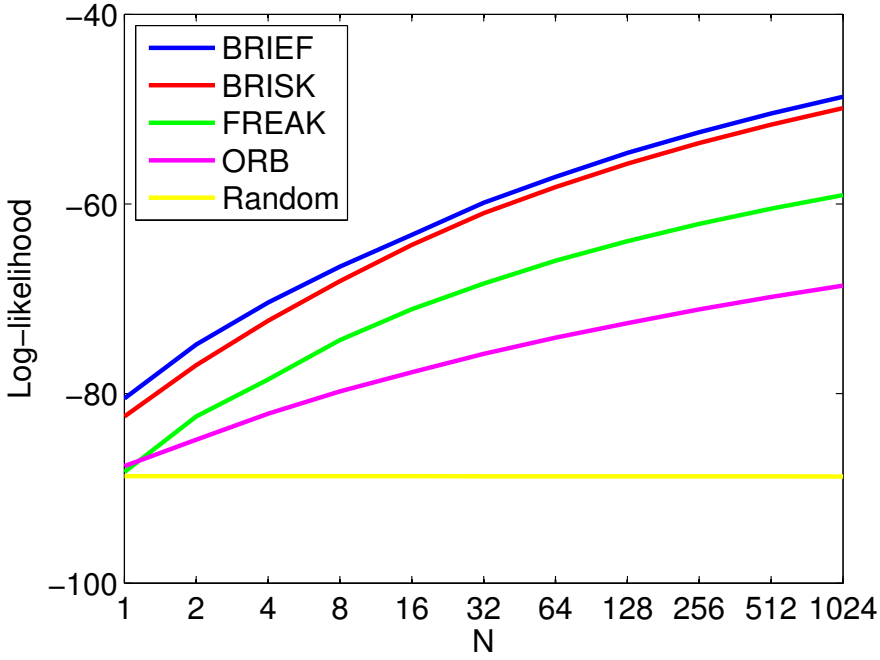
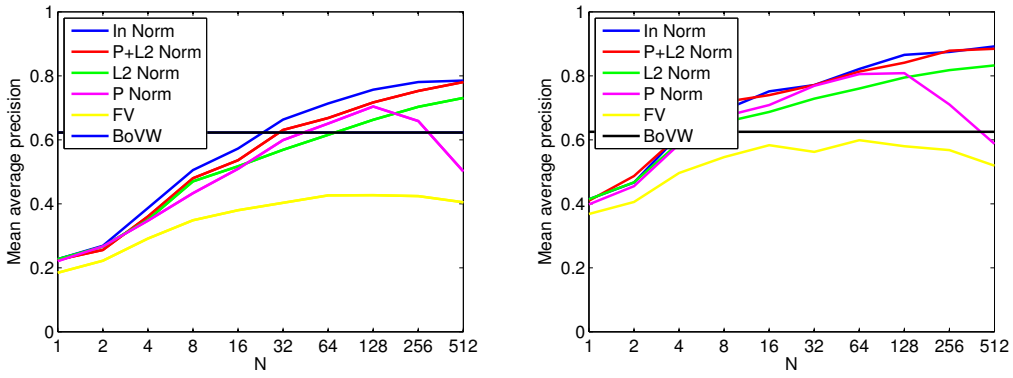
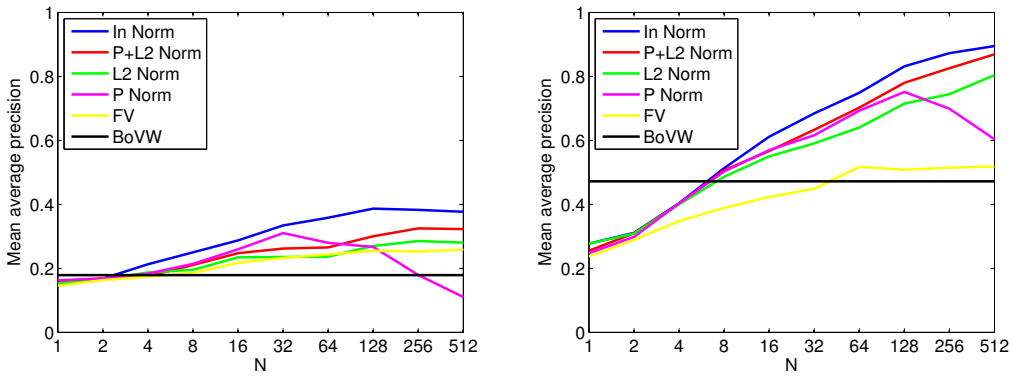


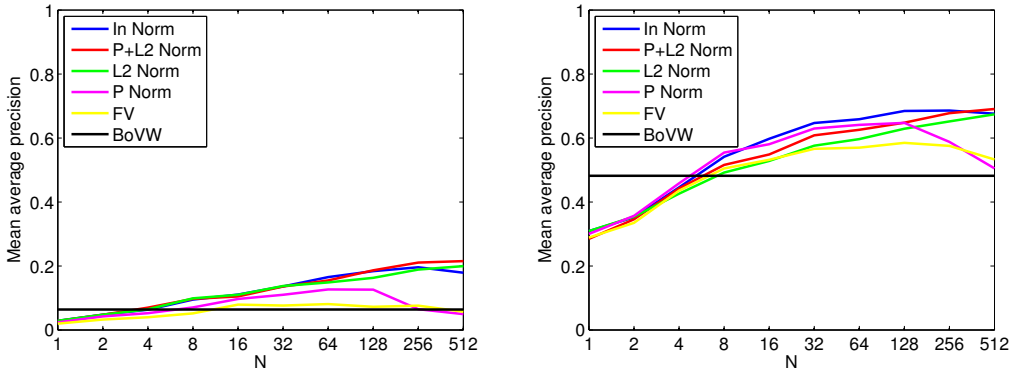
Figure 3.4: Log-likelihood of BMM for different binary features in terms of N.



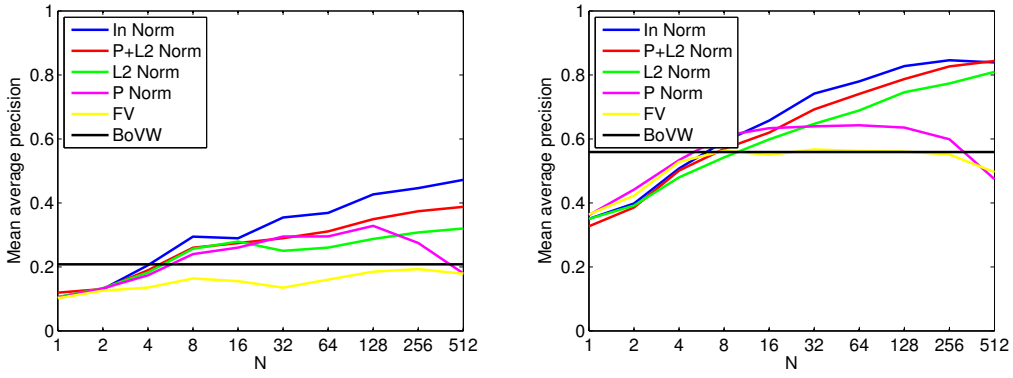
(a) cd (b) book



(c) card (d) dvd



(e) landmark (f) painting



(g) document (h) video

Figure 3.5: Comparison of the Fisher vector and BoBW representations applied to binary features on eight classes

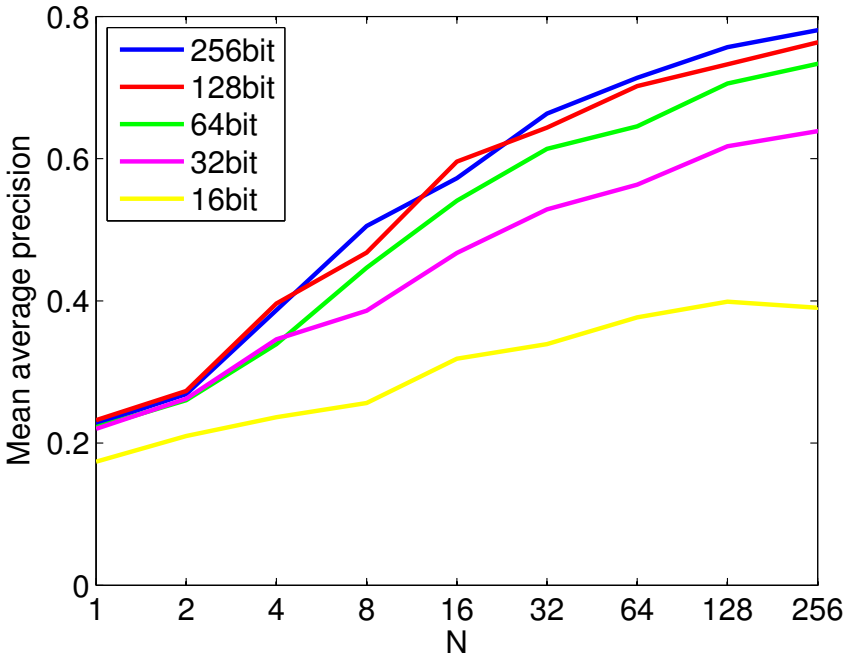


Figure 3.6: Accuracy of the Fisher vector representations under different string lengths.

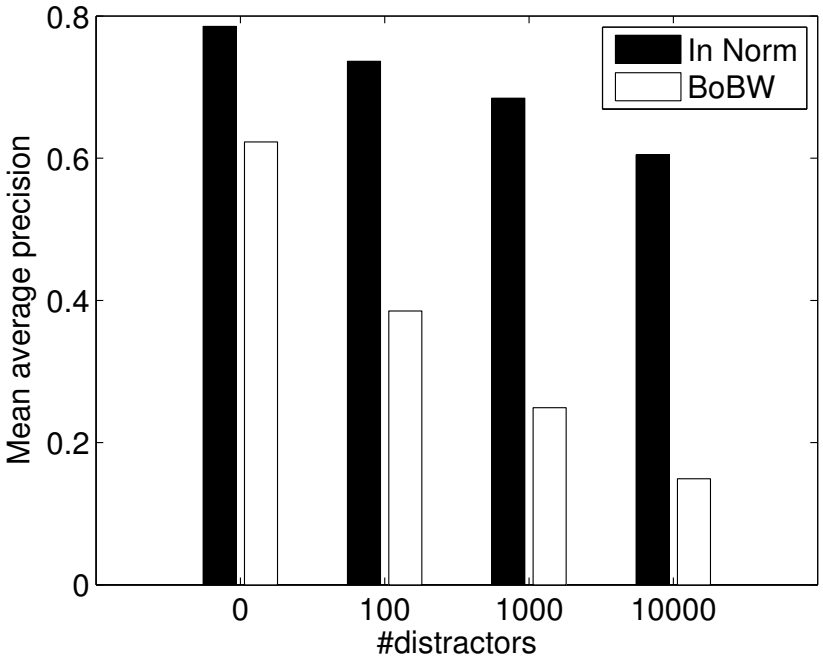
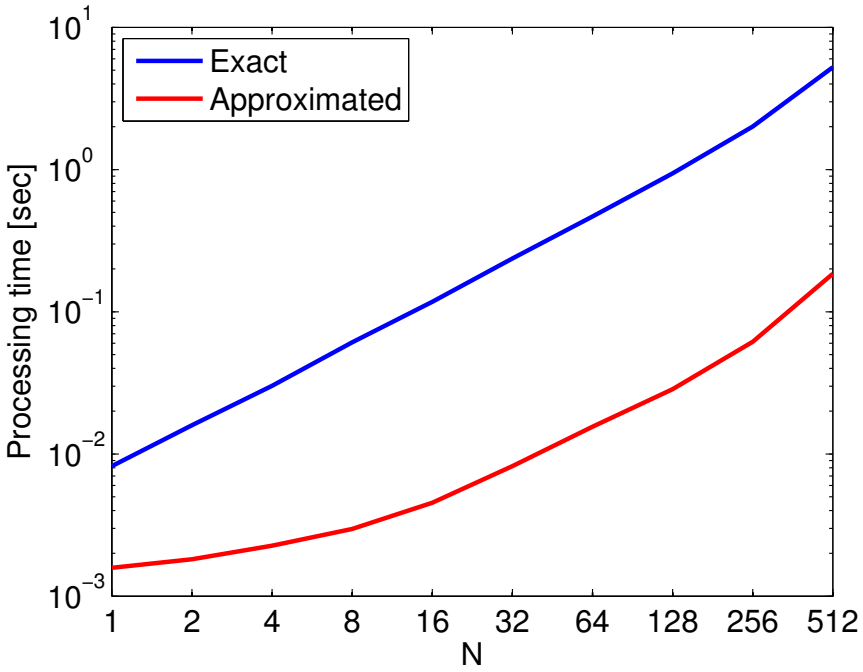
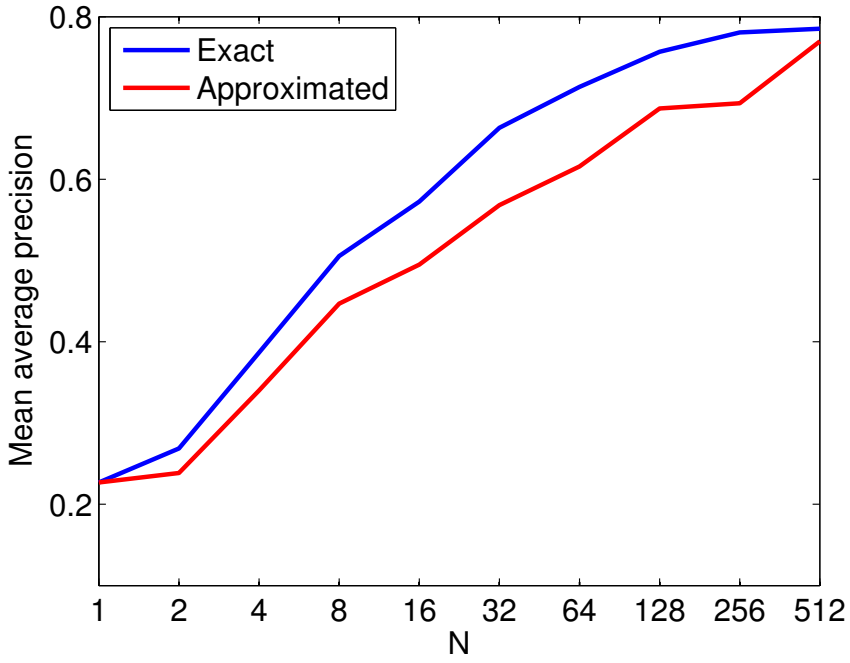


Figure 3.7: Comparison of the Fisher vector and BoBW representations with different numbers of distractors.

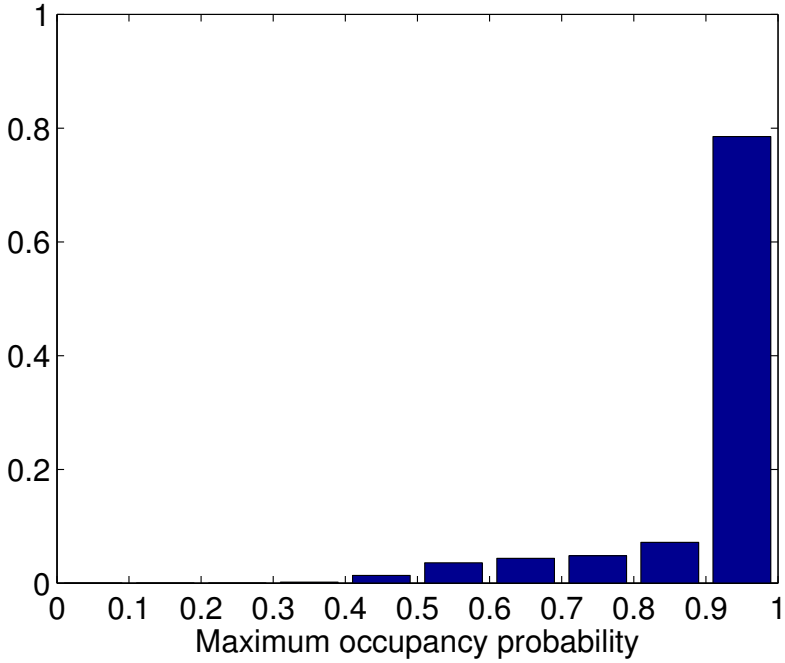


(a) Processing time as a function of N.

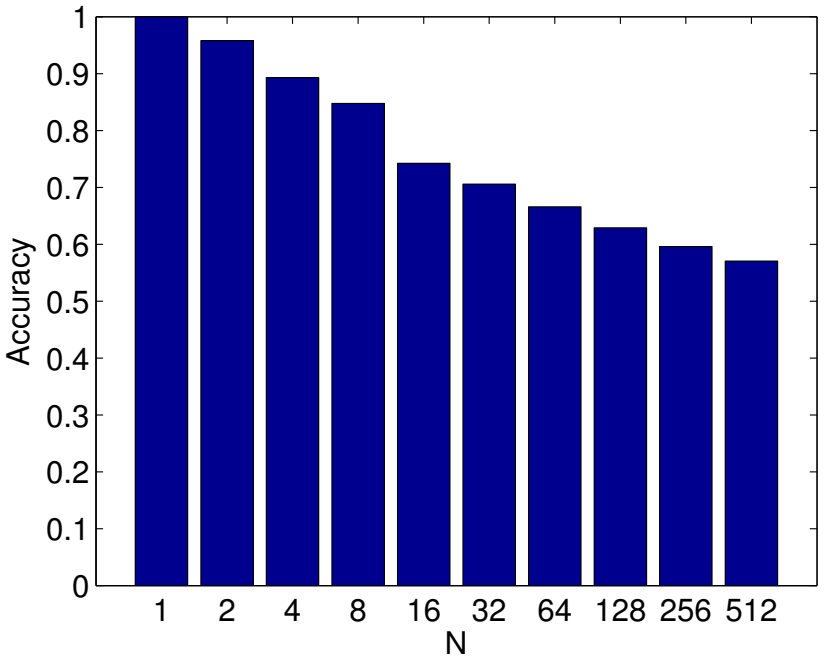


(b) Accuracy as a function of N.

Figure 3.8: Evaluation of fast approximated Fisher vector.



(a) The distribution of maximum occupancy probability $\max_i \gamma_t(i)$ for $N = 512$.



(b) The accuracy of the maximum occupancy probability approximation.

Figure 3.9: The accuracy of two approximations used in the proposed Fisher vector.

Chapter 4

Extended Inverted Index for Binary Features

In this chapter, we propose a stand-alone mobile visual search system based on binary features and the bag-of-visual words framework. The contribution of this chapter is three-fold: (1) we propose a substring extraction method which extracts informative bits from the original binary vector and stores them in the inverted index. These substrings are used to refine visual word-based matching; (2) a modified version of the local naive Bayes nearest neighbor scoring method is proposed in the context of image retrieval, which considers the density of binary features in scoring each feature matching; (3) in order to suppress false positives, we introduce a model check step after standard geometric verification using constraint on the configuration of a transformed reference image. The proposed system improves retrieval accuracy by 11% compared with a conventional method without increasing the database size. Furthermore, our system with the model check does not lead to false positive results.

4.1 Introduction

With the advances in both stable interest region detectors [27] and robust and distinctive descriptors [26], local feature-based image or object retrieval has become a popular research topic. In particular, binary features such as Oriented FAST and Rotated BRIEF (ORB) [19] have attracted much attention due to their efficiency. Binary features are one or two orders of magnitude faster than Scale Invariant Feature Transform (SIFT) [44] or Speeded Up Robust Features (SURF) [47] features in detection and description, while providing comparable performance [19, 30]. These

binary features are especially suitable for mobile visual search or augmented reality on mobile devices[161].

With the increasingly widespread use of mobile devices such as Android phones or iPhones, mobile visual search (MVS) has become one of the major applications of image retrieval and recognition technology. While some research focuses on server-client systems in the context of MVS, the purpose of our research is to achieve fast and accurate recognition with lower memory requirements on mobile devices; in this thesis, we call the latter type of MVS "local MVS". Local MVS does not require any server and it works without a network, realizing faster recognition. Thus, it is suitable for recognizing medium sized databases: i.e., recognizing catalogs, paintings in a museum, or cards in a collectible card game. In this chapter, we assume a real-time local MVS system or application, where images captured by mobile device's camera are continuously used as an input of local MVS system. If the camera captures an object registered in the database, our system is expected to show information or content related to the object immediately. This use case further requires local MVS systems to suppress annoying false positives because objects in the database do not always appear in captured images.

The difficulty for the local MVS lies in indexing of local features because it is necessary to fit the database into the size of the memory in mobile devices or an application while maintaining retrieval accuracy. In other words, managing the trade-off between the memory size of the database and the accuracy of image retrieval is very important. When indexing binary features, Locality Sensitive Hashing (LSH) [174] is often used [19, 161]. However it does not satisfy our constraint because it requires a large amount of memory; original feature vectors and many hash tables should be stored in LSH [82, 175].

A Bag-of-Visual Words (BoVW) framework [61] is the most widely-used approach for local feature-based image or object retrieval that achieves fast retrieval with lower memory requirements. As there is a room for improvement in the accuracy of the standard BoVW framework, many methods have been proposed to improve this framework for continuous features such as SIFT [63, 79, 77, 75]. However, there are not many studies on indexing binary features for the purpose of image retrieval. In [17, 18], the BoVW framework has been adopted to index recent binary features, referred to as Bag-of-Binary Words (BoBW). In [176], a variant of the Hamming embedding method [77] is proposed for binarized features in order to improve the trade-off between memory requirements and accuracy. However, in this method, the quantization process is done by treating the first a bits of a binary feature as an integer

ranging from 0 to $2^a - 1$, and therefore even one bit error alters the quantization result.

In this chapter, in order to achieve a real-time local MVS system, we propose a variant of the BoBW framework, which adaptively extracts and stores VW-dependent information of each binary feature to refine VW-based matching. As the scoring method for matched feature pairs has not been considered in depth and only the standard tf-idf scoring is used in [176], we also propose the use of a modified version of the local Naive Bayes Nearest Neighbor (local NBNN) scoring method [177] for image retrieval, which was originally proposed for image classification. It provides a theoretical basis for scoring feature matching in voting and the proposed modification improves performance by using adaptive density estimation without any additional overhead. Finally, we introduce a geometric verification method in order to suppress false positives.

In this chapter, we did not consider the Fisher vector approach [116, 120, 170] or the Vector of Locally Aggregated Descriptors (VLAD) approach [119, 129, 133], which realizes reasonable retrieval accuracy with very compact image representation. However geometric verification becomes unavailable in these approaches because matching pairs of features are not obtained in search process. Although it is possible to perform feature-level matching after image-level search, which makes these approach not efficient anymore and hurts their advantages.

4.1.1 Extended Inverted Index

Though the BoVW framework achieves efficient retrieval with lower memory requirements, degradation of accuracy is caused by quantization. In the BoVW framework, two features are matched if and only if they are assigned to the same VW [77]. Therefore, quite different features are sometimes matched in the BoVW framework. In order to suppress these unreliable feature matches post-filtering approaches are proposed [77, 82]. In these approaches, after VW-based matching, the distances between a query feature and the reference features that are assigned to the same VW are calculated and matches with large distances are filtered out. In order to perform post-filtering, the inverted index is extended to store additional information of reference features. As exact distance calculation is undesirable in terms of computational cost and memory requirement to store raw feature vectors, short code-based methods are used for this purpose [77, 82]: original feature vectors are encoded into short codes and the distances between feature vectors are approximated by the distances between the short codes. In [77], local feature vectors are encoded

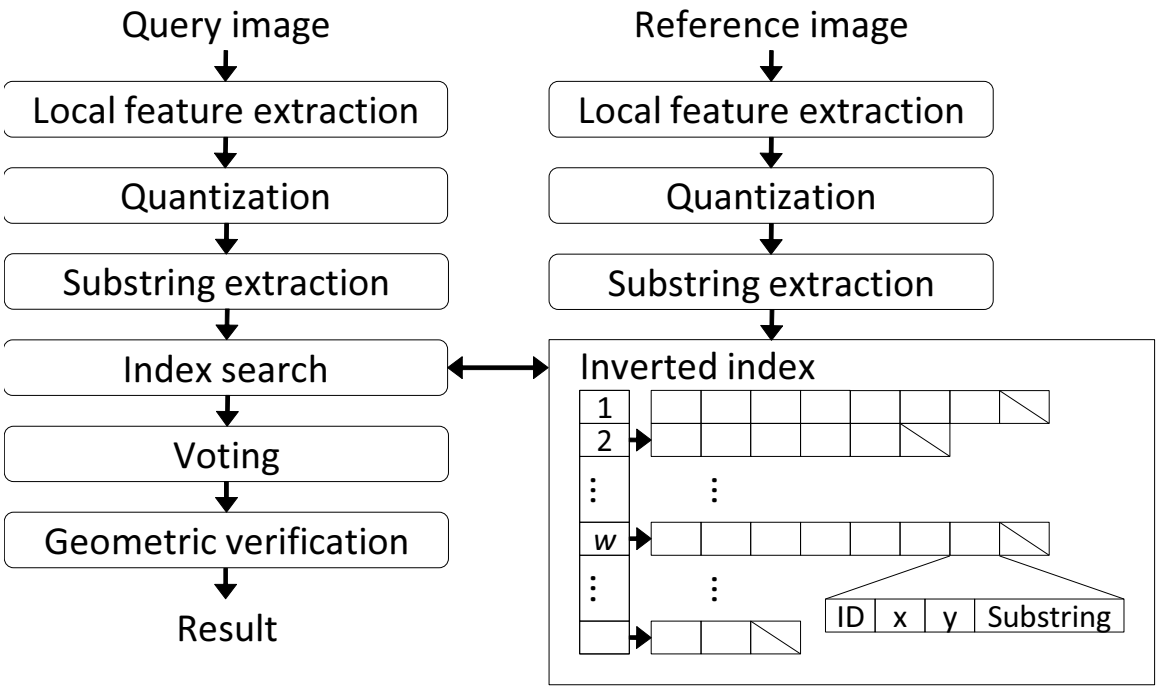


Figure 4.1: The framework of the proposed method.

into binary codes by random projection and thresholding, and resulting binary codes are stored in the inverted index. In [82], a product quantization (PQ) is used to encode reference feature vectors into short codes, and the resulting short codes are also stored in the inverted index.

Although many methods have been proposed to improve the BoVW framework for continuous features, there are not many studies on indexing binary features for the purpose of image retrieval. In [17], the BoVW framework has been adapted to index recent binary features, namely BoBW. In [176], a variant of the Hamming embedding method is proposed for binarized features, where the first a bits are used to form 2^a VWs by treating a bit string as an integer, and the next b-bit substring is stored in an inverted index. However, this approach results in non-optimal performance because directly using binary string as VW is sensitive to bit errors; if one of the first a bits is flipped, the binary vector is to be assigned to a different VW from the original one. Furthermore, as the statistics of binary features are not considered, non-informative bits are sometimes selected as a substring.

4.2 Proposed Local MVS System

In this section, we propose a local MVS system using post-filtering approach, which is suitable for binary features.

4.2.1 Motivation

We first show an observation that was the motivation for our approach. We start with BoBW [17], where VWs are created by clustering full binary vectors, instead of using the first a bits for robustness. The number of VWs is set to 1024. Figure 4.2 shows the statistics of binary features after VW-based clustering. Figure 4.2 (a) and (b) represent mean values of the first 16 bits of the ORB feature assigned to randomly chosen two VWs out of 1024 VWs. Figure 4.2 (c) and (d) represent the correlation among the first 16 bits of the ORB feature assigned to randomly chosen two VWs out of 1024 VWs. Figure 4.2 gives us three observation; (1) the mean values of the binary feature are significantly different from 0.5 at some dimensions, (2) some bits of the binary features are correlated each other, and (3) the characteristics of these correlations and mean values are different from one VW to another. In spite of the first observation above, there are still informative bits after VW-based clustering. Based on this observation, we propose selecting these informative bits as a substring and store the substring in an inverted index for post-filtering.

Figure 4.1 shows the framework of the proposed method, which is almost the same as the BoVW framework or its extensions except for the substring extraction and the inverted index structure. In the indexing step (offline), binary reference features are extracted from reference images and quantized into VWs. The substrings of the reference features are generated and stored in an inverted index to facilitate an efficient search. In the search step (online), each query feature of a query image votes on reference images with scores according to the distances between the query feature substring and reference feature substrings.

4.2.2 Constructing VWs and Substring Dictionary

Before indexing, two training procedures are required in the proposed framework; the construction of VWs and substring dictionary for substring extraction. For VWs construction, the k-means algorithm is first performed on training binary features. Then, the resulting centroid vectors are binarized by thresholding at 0.5 as done in [17], obtaining binary VWs.

Algorithm 1 Substring dictionary construction

Input: Training vectors $\mathcal{X}_w$ assinged to w-th VW, threshold for correlation th
Output: Substring dictionary $\mathcal{D}_w$ with size T

$C \leftarrow$ correlation matrix of $\mathcal{X}_w$
$A \leftarrow$ bit identifiers sorted in ascending order of the distance between the mean value of the bit from 0.5
$\mathcal{D}_w \leftarrow \{A_1\}$
for i **do**2D
    $j \leftarrow A_i$
    if $\max_{k \in \mathcal{D}_w} |C_{jk}| < th$ **then**
        $\mathcal{D}_w \leftarrow \mathcal{D}_w \cup j$
        if $|\mathcal{D}_w| = T$ **then**
            break
        end if
    end if
end for

The other training procedure is the construction of the substring dictionary for bit selection. While the method proposed in [176] uses the fixed positions of bits for the substring, we adaptively change the positions for each VW. For this purpose, we construct substring dictionary $\mathcal{D}_w$, which defines the positions of T useful bits for the w-th VW. For example, in the case where $T = 4$ and $\mathcal{D}_w = \{4, 25, 70, 87\}$, the 4th, 25th, 70th, and 87th bit of each binary feature assigned to the w-th VW are selected, resulting in a 4-bit substring.

The substring dictionary is constructed with the algorithm used in ORB [19], where informative (mean value is close to 0.5) and non-correlated bits are selected. To do this, training vectors are first clustered into N sets $\{\mathcal{X}_w\}_{w=1}^{N}$ using VWs, where N denotes the number of VWs, Algorithm 1 describes the algorithm for the construction of the substring dictionary $\mathcal{D}_w$ for the w-th VW. In this algorithm, bit positions are first sorted according to their entropy. Then, each bit position is added to the dictionary $\mathcal{D}_w$ if the bit is not correlated with all of the bit positions already in $\mathcal{D}_w$. If the algorithm is finished before T bits are selected, the threshold th is decreased and Algorithm 1 is conducted again. The resulting dictionary $\{\mathcal{D}_w\}_{w=1}^{N}$ is used in the following indexing and search step. Storing the dictionary requires $N \times T$bytes because one byte can represent one bit position of a 256-bit string.

4.2.3 Indexing Reference Images

In the indexing step, binary features are extracted from reference images and these features are stored in the inverted index as follows. First, each reference binary feature is quantized into VW. Letting w denote the identifier of quantized VW, the substring is extracted using $\mathcal{D}_w$. Then, the following information is stored in the w-th list of the inverted index as shown in Figure 4.1: image identifier (2 bytes), the position (x, y) (2+2 bytes), and the substring (8 bytes). In total, 14 bytes per feature are required.

4.2.4 Search Step

In the search step, binary features are extracted from a query image. Each query feature votes scores to reference images by the following procedure. First, the binary query feature is quantized into VW w. The substring of the binary query feature is generated in the same manner as in the indexing step using $\mathcal{D}_w$. Then, the distances between the query substring and reference substrings in the corresponding w-th list of the inverted index are calculated. Finally, scores are assigned to the K-nearest neighbor reference features.

4.2.5 Voting Score

It is known that weighting scores according to their distances improves performance. The most common way of doing this weighting is to use the Gaussian function $\exp(-d^2/\sigma^2)$ [79, 91, 77], where d is the Euclidean or Hamming distance between the query feature and reference feature and σ is an adjustable parameter. However, this approach has little theoretical basis and is not optimal. In this chapter, we propose the use of a modified version of the local NBNN (LN) method [177], which has a theoretical background in the derivation of its score. Although LN was originally proposed for image classification, we show that this method also works well in image retrieval. In LN, for each query q, a score of (a) $d_K^2 - d_k^2$ is assigned to the corresponding image of the k-th nearest neighbor feature, where d_x^2 represents the distance between q and its x-th nearest neighbor feature. In this chapter, we modify this original formulation to (b) $(d_K/d_k)^2 - 1$. This modification has the effect of adaptively changing the kernel radius in kernel density estimation like local scaling [178], resulting in more appropriate scoring.

4.2.6 Geometric Verification

Geometric Verification (GV) or spatial re-ranking is important step to improve the results obtained by voting function [63, 70]. In this step, transformations between the query image and the top-R reference images in the list of voting results are estimated, eliminating matching pairs that are not consistent with the estimated transformation. In the estimation, the RANdom SAmple Consensus (RANSAC) algorithm or its variants [93, 94] are used. Then, the score is updated counting only inlier pairs. As a transformation model, an affine or homography matrix is usually used. In our case, we estimate the homography matrix using PROgressive SAmpling and Consensus (PROSAC) algorithm [94] for efficiency. As matching pairs between the query image and the top-R reference images are already obtained in the voting step, these matching pairs are used as an input to RANSAC. In many studies, GV is used only for re-ranking [63, 70] to improve retrieval accuracy. In our case, the purpose of GV is to suppress false positives by thresholding the number of inliers. For this purpose, standard GV is not sufficient; there is an adequate number of inliers for non-relevant image pairs as will be shown in Section 4.3.3.

In this chapter, after standard GV, we check the estimated model using the following constraint, where we assume that reference images represent planar object. Let a, b, c, and d denote the four corners of a reference image in clockwise order. These points are transformed by estimated homography H: $a' = Ha$, $b' = Hb$, $c' = Hc$, and $d' = Hd$. Here, a', b', c', and d' represent the corners of the reference image captured in the query image. Because the angle of each corner of the transformed reference image is never larger than 180 degrees using correct homography, we can use the following constraint to verify the homography: $\overrightarrow{a'd'} \times \overrightarrow{a'b'} > 0$, $\overrightarrow{b'a'} \times \overrightarrow{b'c'} > 0$, $\overrightarrow{c'b'} \times \overrightarrow{c'd'} > 0$, $\overrightarrow{d'c'} \times \overrightarrow{d'a'} > 0$, where $\times$ represents the cross product. If one of the inequalities is not satisfied, the estimated homography is discarded, suppressing false positives in our framework.

Even after the above check, we empirically found some false positives with moderate scores. These were caused by the characteristic of the ORB feature; the multi-scale FAST detector used in ORB tends to detect features at the almost same position (often at corners) with different scales. These features are frequently considered as inliers at the same time, increasing scores between unrelated image pairs.

Figure 4.3 shows real examples of the geometric verification results with the model check. The left images are query, and the right images are reference images. Blue and green lines represent tentative matches before RANSAC. Green lines

Table 4.1: Comparison of the proposed method with conventional methods in terms of substring extraction.

	BoBW [17]	[176]	PROP
book	0.610	0.874	**0.916**
cards	0.173	0.463	**0.535**
cd	0.427	0.752	**0.807**
dvd	0.465	0.811	**0.897**
landmarks	0.080	0.197	**0.253**
paintings	0.486	0.671	**0.718**
text	0.125	0.423	**0.542**
video	0.584	0.824	**0.853**
average	0.369	0.627	**0.690**

represent (false) inliers after RANSAC. Red tetragon represents a reference image transformed by estimated homography. We can see that there are false inliers with almost the same positions. It is also found that false inliers can be clustered into four groups, the minimum number of matches required for RANSAC. In order to reduce this phenomenon, we discard the inliers if are inliers have positions of both reference and query features closer than five pixels, reducing the scores of non-informative matches.

4.3 Experimental Evaluation

In the experiments, the Stanford Mobile Visual Search (SMVS) dataset[1] is used. It contains eight classes of images: camera-phone images of books, business cards, CDs, DVDs, outdoor landmarks, museum paintings, text documents, and video clips. Each class consists of 100 reference images and 400 query images. As an indicator of retrieval performance, mean average precision (MAP; higher is better) [77] is used. We adopt the ORB feature [19] implemented in the OpenCV library[2], where at most 900 features are extracted from four scales on average. The number of VWs is fixed at 1024 in all methods and experiments. The VWs and substring dictionary are trained using the MIR Flickr collection[3].

[1]https://sites.google.com/site/chenmodavid/mobile-visual-search
[2]http://opencv.org/
[3]http://press.liacs.nl/mirflickr/

4.3.1 Effect of Substring Generation

First, the effectiveness of the proposed substring generation method is evaluated. Table 4.1 summarizes the experimental results. The MAP scores of eight classes are shown for three methods: (1) BoBW [17], (2) Zhou's method [176], where we used the first 10 bits to define 1024 (= 2^{10}) VWs and used the next 64 bits as the substring, and (3) the proposed method with 64 bits substring ($T = 64$). For a fair evaluation, conventional tf-idf scoring [44, 179] is used for all methods. Comparing BoBW with Zhou's method, we can see that the use of the substring improves accuracy dramatically. However, the proposed method can further improve accuracy by adaptively generating the substring.

Next, the effect of the number of selected bits and the selection of methods are evaluated. Here, in addition to the proposed method, we evaluate two selection methods: Fixed and Random. Fixed is a modified version of the proposed method, where a substring is created using the first fixed T bits in all visual words. Random uses T bits randomly selected for each visual word. These three selection methods become identical when $T = 256$, where full binary strings of ORB features are always used. Figure 4.4 shows the average MAP scores of eight classes as a function of the length of substrings T, comparing these three selection methods. We can see that the proposed method achieves the best scores among these variants for all T. Fixed is slightly better than Random. This is reasonable because, in the ORB algorithm, binary tests are sorted according to their entropies; the leading bits are more informative. The interesting observation is that the proposed method at $T = 128$ outperforms the proposed method at $T = 256$, while longer bit string achieves a better result using the Fixed and Random methods. This implies that using non-informative or correlated bits degrades search accuracy.

4.3.2 Comparison in Scoring Function

Second, we evaluate the proposed scoring method. The proposed substring method ($T = 64$) with Gaussian weighting (GW) [4] is used as a conventional method. From Table 4.2, it is shown that, while the original LN scoring method (LN(a)) is inferior to GW, the proposed LN scoring method (LN(b)) outperforms GW by 1.5% and the method in [176] by 11% in MAP. This is because scores of original LN are directly affected by the density of feature vectors; the score of a query feature in dense space

[4] For the Gaussian weighting, we set $\sigma = 9$, which achieved the best performance in our preliminary experiments, while $\sigma = 16$ in [91].

Table 4.2: Comparison of the proposed method with conventional methods in terms of scoring.

	PROP+GW	PROP+LN(a)	PROP+LN(b)
book	0.943	0.927	**0.955**
cards	0.602	0.515	**0.609**
cd	0.849	0.830	**0.873**
dvd	0.930	0.924	**0.944**
landmarks	0.278	0.282	**0.289**
paintings	0.740	0.758	**0.773**
text	0.568	0.501	**0.570**
video	0.900	0.909	**0.914**
average	0.726	0.706	**0.741**

Table 4.3: Accuracy of top-1 candidate with and without the model check on eight classes.

	GV	GV+MC
book	0.918	0.918
cards	0.570	0.570
cd	0.838	0.838
dvd	0.930	0.930
landmarks	0.114	0.114
paintings	0.709	0.709
text	0.545	0.545
video	0.868	0.868
average	0.686	0.686

tends be low while the score of a query feature in sparse space tends to be high. The proposed scoring method normalizes the score using the distance between the query feature and its K-th nearest neighbor feature in the database. Thus, all query features can equally contribute to the similarity score, improving the final result. As the overhead of the proposed method is negligible, the proposed system can improve retrieval accuracy with the same memory requirements and almost the same computational cost as conventional methods.

4.3.3 Results with Geometric Verification

The accuracy of the proposed framework with geometric verification is evaluated. In this chapter, we perform geometric verification against the top-3 results of LN(b). Firstly, we evaluate the effectiveness of the model check method described in

Section 4.2.6 against false positive suppression. For this purpose, we conducted an experiment where cd class is used as the reference, and 10,000 images distinct from the cd class are used as as queries. In this case, ideally, the scores of all results should be zero. Figure 4.5 shows the distributions of the top-1 score obtained by standard geometric verification (GV) and with the model check described in Section 4.2.6 (GV+MC). We can see standard geometric verification returns relatively high scores despite the fact that there is no relevant image in the database for any of the queries. In contrast, the geometric verification with the model check can suppress most of the false positives. In particular, there is no result with a score larger than eight. Therefore, if we use a threshold larger than eight, we can achieve a system with no false positives.

Table 4.3 shows the top-1 accuracy of GV and GV+MC. Compared with LN(b) in Table 4.2, Accuracy declines because geometric verification sometimes discards true positives without a large number of inliers. However, the model check does not cause further degradation of the accuracy. The degradation of accuracy caused by GV is prominent in the landmarks and painting classes. The landmark class includes many non-planar objects, and GV based on homography sometimes fails. The painting class includes less textured images, lacking in enough inliers in RANSAC.

4.3.4 Computational Cost and Memory Requirements

Table 4.4 shows processing times for feature extraction (Feature), quantization (Quantize), Hamming distance calculation and voting (Hamming), geometric verification (GV). These durations are measured on a standard PC with a Core i7 2600 CPU 3.4GHz and 32 GB of main memory (PC), and on an IS11S smartphone (released in 2011) with Qualcomm Snapdragon S2 MSM8655 1 GHz (SP). Extracting substrings is quite simple and thus fast, its computational cost is negligible. The times required for Feature and GV take significantly longer on the smartphone, while those for Quantize and Hamming do not. This is because, in our implementation, Quantize and Hamming processes can be sped up using the ARM NEON SIMD operation on the smartphone. We can see that recognition rates are about 14 fps on the PC and about 2 fps on the smartphone, achieving a real-time local MVS with no false positives. Comparing our method and Zhou's method [176], Zhou's method is a little faster because Zhou's method does not require the quantization process in Table 4.4. However, the above acceleration results in non-optimal performance in terms of search precision as discussed in Section 4.1.1 and shown in Table 4.1.

Table 4.4: Processing times [sec] of the proposed MVS system.

	Feature	Quantize	Hamming	GV	Total
PC	0.009	0.015	0.013	0.035	0.072
SP	0.184	0.076	0.053	0.211	0.524

In terms of memory requirements, we assume that 100 images are indexed, 900 features are extracted per image, the number of VWs is 1024, and the length of substring is 64. Under this settings, 100×900×14 bytes = 1.26 Mbytes is required to store features in inverted index, 1024×32 bytes $\simeq$ 0.03 Mbytes for VWs, and 1024×64 bytes $\simeq$ 0.07 Mbytes for the substring dictionary. This amount of data can reasonably be included in application binary files such as .apk for Android or .ipa for iPhone. Comparing our method and Zhou's method [176], Zhou's method does not need to store VWs (0.03MB) and the substring dictionary (0.07MB). This memory footprint is relatively small compared with that of the inverted index and it does not increase even if the number of images is increased.

4.3.5 Toward Large-scale Image Retrieval System

While our system focuses on the local mobile visual search applications, we'll show that our system can be applicable to large-scale image retrieval applications. For this purpose, we extend our basic system proposed in this Chapter using the following approaches:

- Large vocabulary: we here use a relatively large vocabulary (up to 80K) for efficiency. If a small size of vocabulary is used, the length of the lists in the inverted index becomes too large for a large-scale dataset, which increases the processing time for the Hamming distance calculation. On the other hand, a large vocabulary increases the processing time for quantization. Therefore we adopt a hierarchical quantization approach [62, 77] with a two-layered codebook. In the codebook construction, the lower codebook is first created with the k-means algorithm, and then the upper codebook is created by further clustering the centroids of the lower codebook. Letting N denote the size of the lower codebook, we set the size of the upper codebook to $\sqrt{N}$.

- Weak geometric verification: WGC improves the image retrieval accuracy at the cost of a small increase of memory requirements. We adopt orientation-based WGC because WGC based on scale is less useful than one based on the orientation as shown in [102].

- Feature selection: selecting useful features is very important for reducing the size of the database or improving the search accuracy at the same memory consumption [180–182]. We adopt our feature selection method [182] that generates multiview synthetic images and extracts local features, and then these features are resampled according to a reliability measure.
- Other tunings: we extract the ORB features from 8 scales instead of 4 scales. This slightly increases the accuracy with a small increase of the extraction time. While feature matchings for GV are obtained by performing exhaustive matching for each pair of images for high repeatability in the previous experiments, here we use the matching pairs obtained in voting for efficiency.

As reference images, we use the book, CD, and DVD classes, which are most appropriate test objects for our use cases explained in Section 4.1. In addition to the above reference images, we add one million distractor images from the MIR Flickr collection in order to confirm the scalability of the proposed system.

Table 4.5 shows the experimental results using this large-scale dataset. Several different parameter settings are used: the size of the codebook N, the length of the substring T, and the number of features for a reference image. For query images, 900 features are extracted irrespective of the number of features for the reference image. For accuracy, average MAP for the three classes is shown. We can see that the parameters have a large impact on the results.

The size of the codebook. Due to the hierarchical structure of the quantization and the SIMD optimization, the size of the codebook does not have a large impact on the processing time for the quantization. On the other hand, using a small size of codebook (N = 40K) increases the processing time for the Hamming distance calculation. This is because the length of a inverted index list increases by two times on average if the size of the codebook is reduced to halve. A larger size of the codebook degrades the accuracy because the recall of the feature-level matching declines.

The length of the substring. The longer substring length T drastically improves the accuracy while increasing the size of the database. It also increases the processing time for the Hamming distance calculation but not linearly to the length of the substring.

The number of features per reference image. A larger number of reference features increases the computational cost for the Hamming distance calculation linearly and increases the size of database. In this experiment, the number of the reference features does not have much impact on the accuracy.

Table 4.5: The results of the large-scale experiments. The MAP score and the processing times [sec] of each step and total querying time are shown for each parameter setting.

Parameters	N	40K	80K	80K	80K
	T	64	64	128	128
	#Features	900	900	600	900
Timings	Feature	0.015	0.015	0.015	0.015
	Quantize	0.007	0.007	0.007	0.007
	Hamming	0.161	0.077	0.064	0.092
	GV	0.001	0.001	0.001	0.001
	Total	0.184	0.100	0.087	0.115
MAP		0.725	0.563	0.835	0.857

Table 4.6: Summary of the impacts of the parameters. The character '+' represents 'increase' of the corresponding parameter or performance measure.

	Quatize	Hamming	MAP
N+	+	- - -	- -
T+		++	+++
#Features+		+++	++

Table 4.6 summarizes the impacts of the parameters. The character '+' and '-' respectively represent 'increase' and 'decrease' of the corresponding parameter or performance measure.

From the experiments, we adopt the parameter set N = 80K, T = 128, and #Fetures = 600 for our large-scale retrieval system as a good solution of the trade-off among accuracy, memory requirements, and processing time. Finally, Figure 4.6 shows MAP and processing time of the system as a function of the number of the distractor images. We can see that the accuracy declines slowly rather than linearly as the number of the distractor images increases. The processing time is linearly increases against the number of the distractors. This is because the processing time for the Hamming distance calculation becomes dominant in this setting as implied in Table 4.5. This increase would be alleviated if a larger size of the codebook is used.

4.3.6 Comparison with the other local MVS framework

Finally, we compare our system with the state-of-the-art system in the area of the local MVS proposed by Hartl et al. [16]. Table 4.7 shows the comparison of our system and the Hartl's system. For MAP, the average of eight classes is shown.

Table 4.7: The comparison of our system and the Hartl's system.

	Ours	[16]
Image resolution	640	320
Local feature	ORB	PCA-SURF
# of features	900	256
Processor	SnapDragon 600 1.51 GHz	Cortex-A9 1.4 GHz
Processing time [ms]	361	376
Feature extraction	243	216
Quantization	(18K/2 layers) 65	(1M/6 layers) 26
Matching	44	-
Geometric verification	9	134
Memory [MB]	17.8	21.6
Recognition rate	0.811	0.700

While the accuracy is calculated for each class of the SMVS dataset in the previous experiments, the images of all classes are used as reference images in this experiment and recognition rate is used as a performance measure (top-1 accuracy) as done in [16]. For the Hartl's system, the values are taken from [16]. While there are many differences between our system and the Hartl's system, the biggest difference is that our system utilizes the binary feature, ORB, and the Hartl's system uses the non-binary feature, SURF. Because non-binary features are one order of magnitude slower than binary features, systems based on non-binary features requires tunings. The Hartl's system restricts the maximum height or width of images to 320 pixels and the maximum number of local features to 256. Although this reduces the processing time for feature extraction, it degrades the final accuracy at the same time. The processing time for quantization on the Hartl's system is shorter than that of ours because the Hartl's system uses the vocabulary tree [62]. We think that the vocabulary tree is not suitable for a local MVS system (or a small-scale database) because it requires a large memory, 11.1 MB for the Hartl's system. The Hartl's system also stores local features independent of the inverted index for geometric verification, which increases the processing time for geometric verification and memory requirements. From Table 4.7, we can see that our system significantly outperforms the Hartl's system in terms of MAP. We think that this comes from the tunings done in order to speed up extracting the SURF feature as mentioned before.

4.4 Summary

In this chapter, we proposed a stand-alone mobile visual search system based on binary features. In our system, a VW-dependent substring extraction method and a new scoring method are used. It is shown that the proposed system can improve retrieval accuracy with the same memory requirement as conventional methods. Geometric verification using a constraint on the configuration of a transformed reference image achieved no false positive results. Furthermore, we demonstrated the scalability of our system using up to one million database images. Finally, we compared our system with one of the state-of-the-art systems in details and showed the

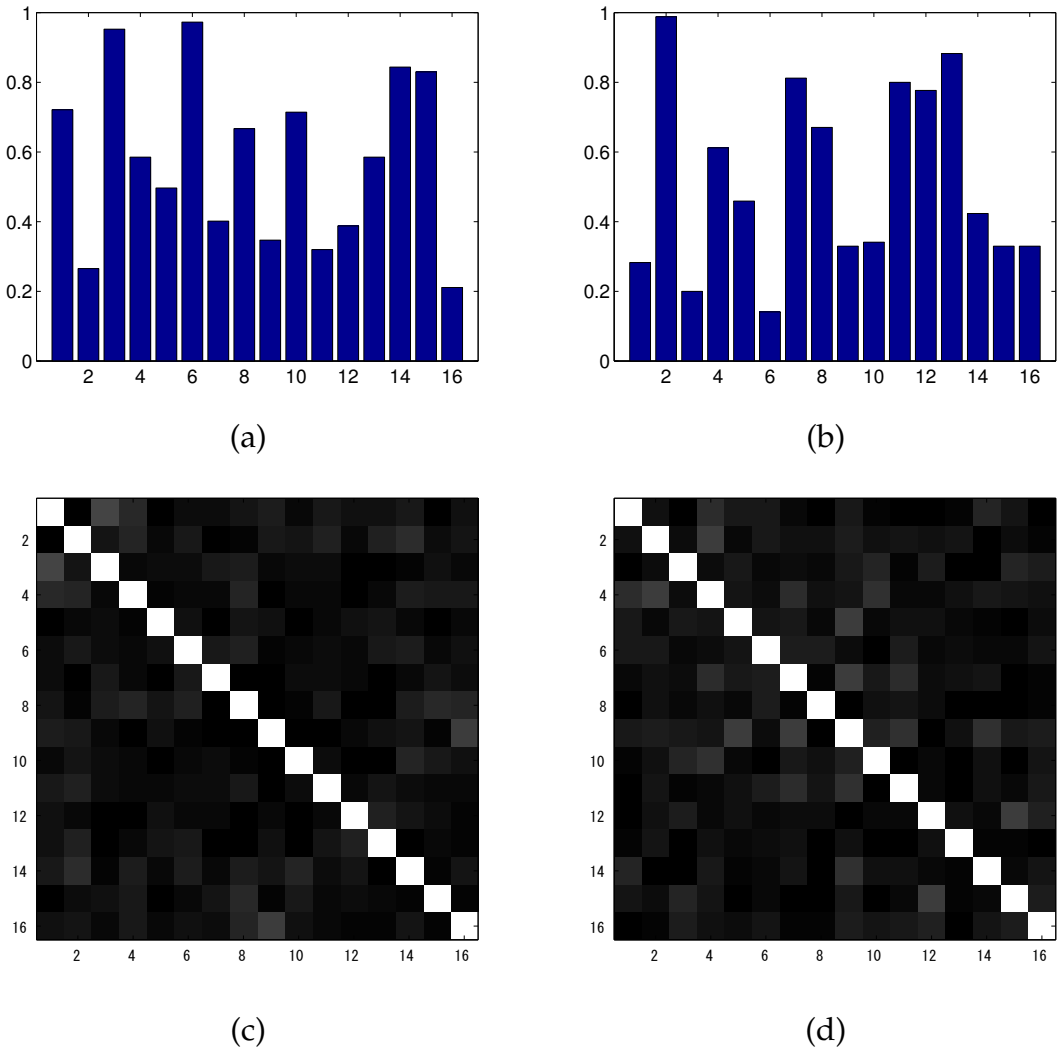
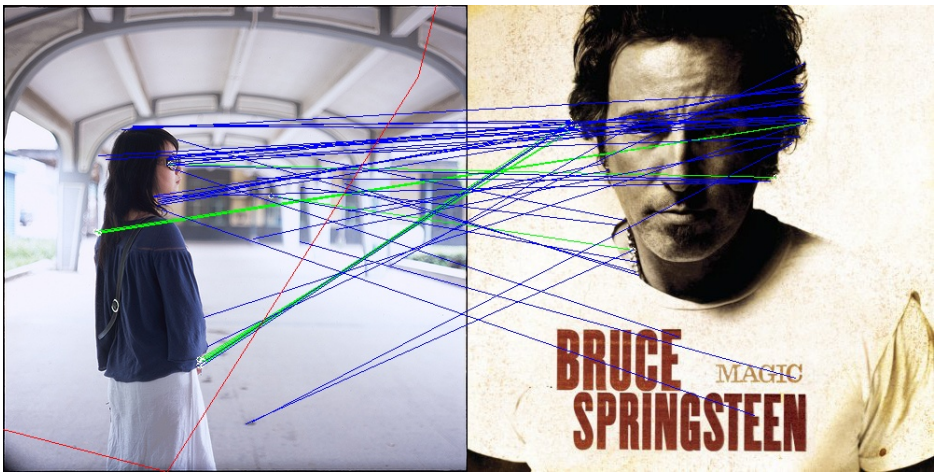
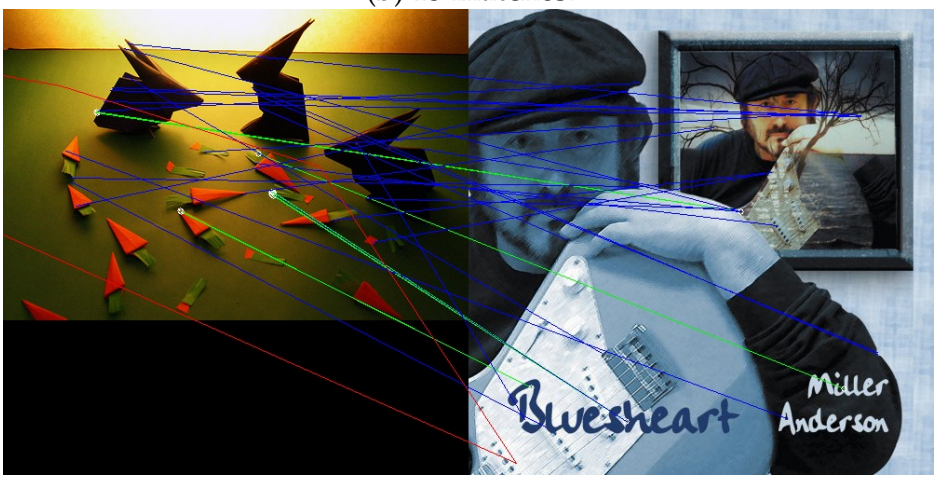


Figure 4.2: (a), (b): mean values of first 16 bits of ORB feature assigned to randomly chosen two VWs. (c), (d): correlation among first 16 bits of ORB feature assigned to randomly chosen two VWs.



(a) 14 matches.

(b) 13 matches.



(c) 11 matches.

Figure 4.3: Geometric verification results with the model check. The left images are query, and the right images are reference images. Blue and green lines represent tentative matches before RANSAC. Green lines represent (false) inliers after RANSAC. Red tetragon represents a reference image transformed by estimated homography.

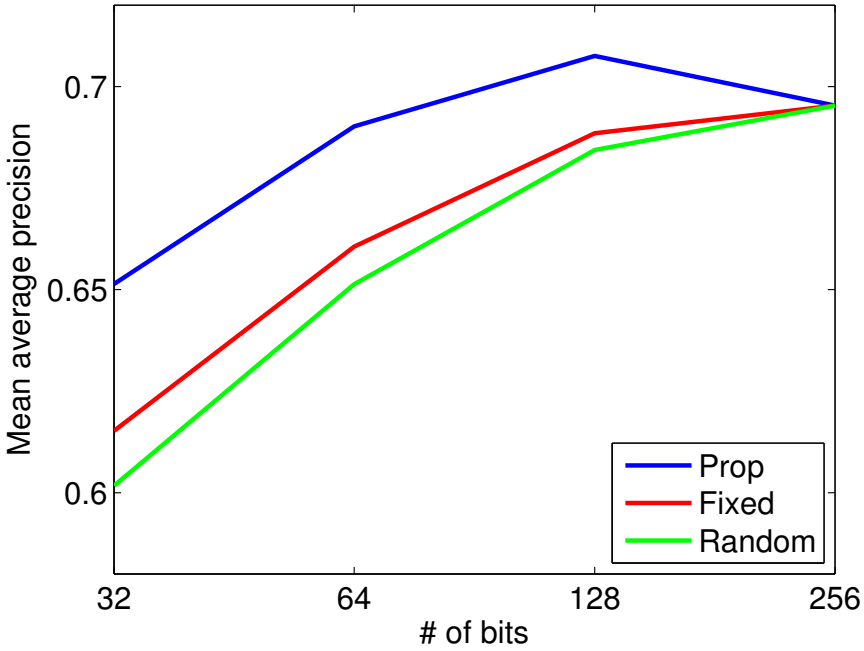


Figure 4.4: Average MAP scores of eight classes as a function of the length of substrings T.

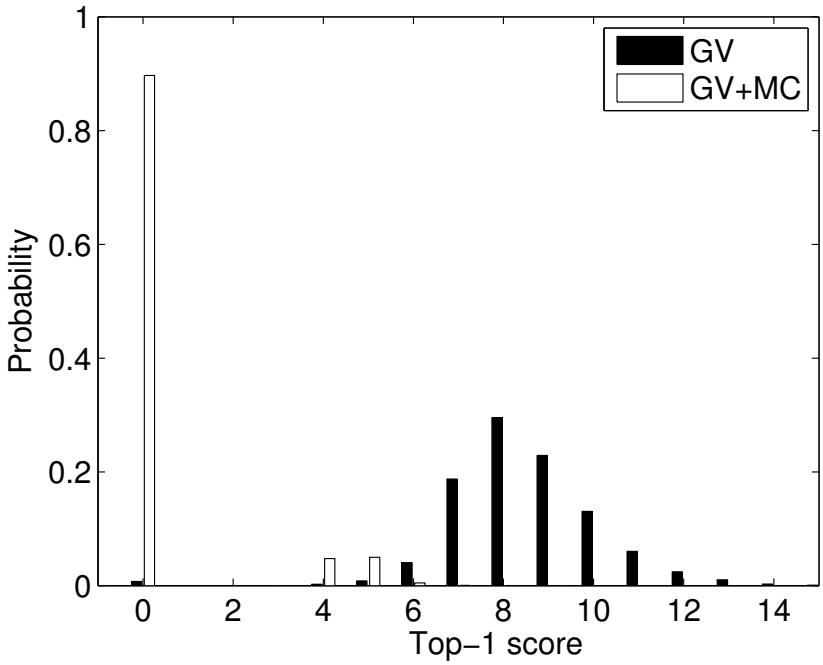


Figure 4.5: Distribution of the top-1 scores of 10,000 non-relevant queries with and without the model check.

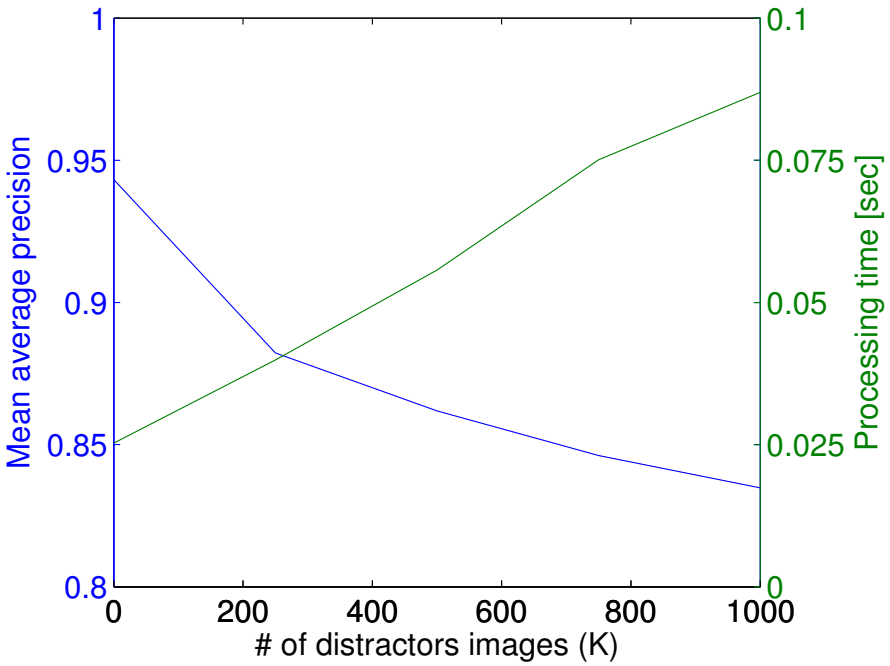


Figure 4.6: MAP and processing time as a fucntion of the number of the distractor images.

Chapter 5

Linking Fisher Kernel to Inverted Index-based Systems

In this chapter, we propose to integrate the advantages of the approaches proposed in Chapter 3 and Chapter 4. Although the Fisher vector approach proposed in Chapter 3 provides good similarity measurement to binary features, the resulting Fisher vector representation does not satisfy our requirements because geometric verification becomes unavailable in this approach; the matching pairs of features are not obtained in search process. While the extended inverted index approach proposed in Chapter 4 can realize a practical MVS system, there is room for improvement in the distance calculations involved in the quantization and voting. Starting with general match kernel, we show that the Fisher kernel-based similarity measurement can be implemented using the extended inverted index structure.

5.1 Proposed Approach

In this section, we introduce Fisher match kernel for binary features, deriving two different search methods. We start with general match kernel. Let us consider two sets $\mathcal{X}$ and $\mathcal{Y}$ such that $|\mathcal{X}| = n$ and $|\mathcal{Y}| = m$. Each set consists of a set of vectors, such as local descriptors associated with an image. We first consider match kernels, in a framework derived in the literature [183], that have the form:

$$K(\mathcal{X}, \mathcal{Y}) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} k(x, y). \tag{5.1}$$

Selecting appropriate kernel is very important for final results. Many methods which involves matching of binary features use Hamming distances to measure dissimilarities between two binary features [77, 19, 92]. These methods do not consider distributions of binary feature vectors, and implicitly assume that each bit is independently generated and its mean value is 0.5. However, as shown in experimental results, these assumptions do not hold for binary features. In this section, we adopt Fisher kernel which can consider underlying distributions of binary feature vectors. It is natural to use Fisher kernel with the success of Fisher vectors in image classification [114, 116] or image retrieval [118]. In the following, we first model binary feature vectors with Bernoulli mixture model (BMM). Then, it will be shown that the Fisher kernel of BMM can be considered a variant of the BoVW framework.

5.1.1 Bernoulli Mixture Model

In modeling binary features, it is straightforward to adopt a single multivariate Bernoulli distribution. However, although many binary descriptors are designed so that bits of resulting binary features are uncorrelated [19], there are still strong dependencies among bits. Therefore, a single multivariate Bernoulli component will be inadequate to cope with the kind of complex bit dependencies that often underlie binary features. This drawback is overcome when several Bernoulli components are adequately mixed. In this chapter, we propose to model binary features with a multivariate Bernoulli mixture (BMM). The use of the BMM instead of a single multivariate Bernoulli distribution will be justified in the experimental section.

Let $x \in \{0,1\}^D$ denote a D-dimensional binary feature and $\lambda = \{w_i, \mu_{id}, i = 1..N, d = 1..D\}$ denote a set of parameters for a multivariate Bernoulli mixture model with N components, and x_d represents the d-th bit of x.

$$
\begin{aligned}
p(x) &= \sum_{i=1}^{N} w_i p_i(x), \\
p_i(x) &= \prod_{d=1}^{D} \mu_{id}^{x_d} (1-\mu_{id})^{1-x_d}.
\end{aligned} \tag{5.2}
$$

In order to estimate the values of the parameter set λ, given a set of training binary features $x_1, \cdots, x_s, \cdots, x_S$, the expectation-maximization (EM) algorithm is applied [173]. In the expectation step, the occupancy probability $\gamma_s(i)$ (or posterior

probability $p(i|x_s, \lambda)$) of x_s being generated by the i-th component of BMM is calculated as

$$p(i|x_s) = \frac{w_i p_i(x_s|\lambda)}{\sum_{j=1}^{N} w_j p_j(x_s|\lambda)}. \tag{5.3}$$

In the maximization step, the parameters are updated as

$$S_i = \sum_{s=1}^{S} p(i|x_s),\ w_i = S_i/S,\ \mu_{id} = \frac{1}{S_i}\sum_{s=1}^{S} p(i|x_s)x_{sd}. \tag{5.4}$$

5.1.2 Fisher Kernel of the BMM

In this section, we derive the Fisher kernel of the BMM. The Fisher kernel between two binary features x and y is defined as:

$$k(x, y) = G^x F_\lambda^{-1} G^y, \tag{5.5}$$

where G^x and G^y are the Fisher scores and F_λ is the Fisher information matrix. These terms are calculated in a similar way as done in Chapter 3. The difference from Chapter 3 is that we consider the Fisher kernel between only two feature vectors while the two sets of feature vectors are used in Chapter 3. Using similar derivation, we obtain:

$$\begin{aligned} G^x_{\mu_{id}} &= \frac{\partial \mathcal{L}(x|\lambda)}{\partial \mu_{id}} \\ &= \frac{1}{p_i(x|\lambda)} \frac{\partial p_i(x|\lambda)}{\partial \mu_{id}} \qquad (5.6) \\ &= p(i|x) \frac{(-1)^{1-x_d}}{\mu_{id}^{x_d}(1-\mu_{id})^{1-x_d}}, \qquad (5.7) \end{aligned}$$

and

$$F_{\mu_{id}} = w_i \left(\frac{\sum_{j=1}^{N} w_j \mu_{jd}}{\mu_{id}^2} + \frac{\sum_{j=1}^{N} w_j (1-\mu_{jd})}{(1-\mu_{id})^2} \right), \tag{5.8}$$

where $G^x_{\mu_{id}}$ and $F_{\mu_{id}}$ denote diagonal elements of G^x and F_λ w.r.t. the parameter μ_{id}. Substituting $G^x_{\mu_{id}}$ for Eq. (5.5), we get:

$$k(x, y) = \sum_{i=1}^{N} \left(p(i|x) p(i|y) \alpha_i(x)^\top F_i^{-1} \alpha_i(y) \right), \tag{5.9}$$

where $\alpha_i(x)$ is a vector whose d-th component is:

$$\alpha_{id}(x) = \frac{(-1)^{1-x_d}}{\mu_{id}^{x_d}(1-\mu_{id})^{1-x_d}}, \tag{5.10}$$

and F_i is a diagonal matrix whose (d,d) component is $F_{\mu_{id}}$. As $p(i|x)$ is very peaky, we can use the following approximation:

$$p(i|x) = \begin{cases} 1 & \arg\max_j p(j|x) = i \\ 0 & \textit{otherwise}. \end{cases} \tag{5.11}$$

Using this approximation, $k(x,y)$ in Eq. (5.9) becomes 0 if $\arg\max_i p(i|x) \neq \arg\max_i p(i|y)$. If $\arg\max_i p(i|x) = \arg\max_i p(i|y) = \hat{i}$, $k(x,y)$ has non-zero value:

$$k(x,y) = \begin{cases} \alpha_{\hat{i}}(x)^\top F_{\hat{i}}^{-1} \alpha_{\hat{i}}(y) & \hat{i} = \arg\max_j p(j|x) \\ & = \arg\max_j p(j|y) \\ 0 & \textit{otherwise}. \end{cases} \tag{5.12}$$

We derive two proposed methods from this kernel, namely BMM-VW and BMM-FK.

5.1.3 BMM-VW

In Eq. (5.12), similarity between x and y becomes 0 if $\arg\max_i p(i|x) \neq \arg\max_i p(i|y)$ as discussed above. This is very similar behavior to the BoVW framework, where x and y does not contribute to similarity if they are quantized into different VWs. Namely, we can regard multivariate Bernoulli components as VWs, and consider the calculation of $\arg\max_i p(i|y)$ as quantization process. Thus, we propose the BMM-VW framework, where the following kernel function is used in Eq. (5.1):

$$k(x,y) = \begin{cases} 1 & \arg\max_i p(i|x) = \arg\max_i p(i|y) \\ 0 & \textit{otherwise}. \end{cases} \tag{5.13}$$

In this formulation, two feature vectors x and y themselves are not required to calculate the kernel value; only $\arg\max_i p(i|x)$ and $\arg\max_i p(i|y)$ are required. Therefore, we can use the inverted index data structure for efficient similarity search.

5.1.4 BMM-FK

In this section, we introduce *selectivity* to the kernel in Eq. (5.12). In this kernel, x and y are considered to be matched if $\arg\max_i p(i|x) = \arg\max_i p(i|y)$, which correnponds to the situation that x and y are quantized into the same VW in the context of the BoVW framework. In [64, 77], it is shown that this matching cause many false positives. In order to suppress these false positives, it is proposed that matched pairs are further filtered out according to the distance or similarity between two feature vectors [64, 77, 83, 110, 184]. For example, in the Hamming embedding [77] framework, a matched pair x and y is filtered out if their Hamming distance is larger than a threshold. The selective Fisher kernel is defined as:

$$k(x,y) = \begin{cases} \rho\left(\alpha_{\hat{i}}(x)^\top F_{\hat{i}}^{-1} \alpha_{\hat{i}}(y)\right) & \hat{i} = \arg\max_j p(j|x) \\ & = \arg\max_j p(j|y) \\ 0 & \textit{otherwise}. \end{cases} \tag{5.14}$$

The selectivity function ρ is, for example, defined as:

$$\rho(u) = \begin{cases} u & u \geq \tau \\ 0 & \textit{otherwise}. \end{cases} \tag{5.15}$$

This function discards the matched pairs according to similarity (kernel value) of the matched pair. However, it is shown that absolute value of distance or similarity is not appropriate for thresholding [83, 185]. Therefore, it is proposed to match each query feature with only the top-k features in the database. In this chapter, we also adopt this strategy in selectivity.

In indexing, we can adopt the same extended inverted index data structure proposed in Chapter 4 (Figure 4.1). The differences from the framework in Chapter 4 is the quantization step and the voting step. In quantization, BMM-FK utilizes the components of BMM as codebook. In voting, the kernel value defined in Eq. (5.12) is used as a score, which is voted to the top-k features.

5.1.5 Fast Posterior Calculation with Randomized BMM Trees

The most time consuming part in both BMM-VW and BMM-FK is the calculation of posterior $p(i|x) = \frac{w_i p_i(x)}{\sum_{j=1}^N w_j p_j(x)}$. More precisely, given a set of parameters of BMM $\lambda = \{w_i, \mu_{id}, i = 1..N, d = 1..D\}$ and a feature vector x, the identifier of BMM component

with the largest posterior $\hat{i} = \arg\max_i p(i|x)$ should be obtained in Eq. (5.13) or Eq. (5.14).

Similarly, in the BoVW framework, it is required to find the nearest neighbor VW from x in quantization. This is exactly a Nearest Neighbor Search (NNS) problem in Euclidean distance [74, 186–188, 82], and therefore algorithms for NNS can be used. In practice, approximate NNS algorithms are frequently adopted for efficiency. In [62], a hierarchical tree structure called a vocabulary tree is used for efficient quantization. In [63], randomized kd-trees are adopted, which is shown in [188, 189] to outperform the other approximate NNS methods such as Locality Sensitive Hashing (LSH) or hierarchical k-means tree.

Backing to our case, the above NNS algorithms are not applicable because (1) the input x is a binary vector, and data to be searched is BMM components; (2) the distance between x and a BMM component is defined by (log-)likelihood. Therefore, we propose to use the forest of Geometric Near-neighbor Access Trees (GNATs) [190], which hierarchically decomposes the search space by randomly selected data points and works for any metric distance. In [191, 175], similar NNS algorithms for binary features are proposed, which perform priority search of multiple hierarchical clustering trees. Although we also use multiple trees (forest) and priority search, the algorithm is different because the input x is a binary feature, and data to be searched is also binary features in [191, 175].

Algorithm 2 Constructing a randomized BMM tree

Input: BMM components B
Output: randomized BMM tree

if $|B| < S_L$ **then**
  create leaf node with Bernoulli components B
else
  $P_1, \cdots, P_K \leftarrow$ select K Bernoulli components at random from B
  $C_1, \cdots, C_K \leftarrow$ cluster Bernoulli components in B around Bernoulli components minimizing KL divergence
  for each cluster $C_i \in C$ **do**
    create non-leaf node with P_i recursively apply the algorithm to C_i
  end for
end if

Constructing Randomized BMM Trees

Algorithm 2 shows the algorithm to construct a randomizedl BMM tree. It starts with all input BMM components and they are divided into K clusters, where K is a parameter of the algorithm, called the branching factor. The clusters are formed by first selecting K BMM components as centroids at random (Line 4), and then each of the other BMM components is assigned to the nearest BMM component among the randomly selected K BMM components (Line 5). Randomly selecting centroids instead of using clustering algorithms [62, 188] ensures independency among multiple BMM trees. The algorithm is repeated recursively for each of the resulting clusters until the number of points in each cluster is below a certain threshold, the maximum leaf size parameter S_L. This algorithm is repeated to obtain T BMM trees.

In the above algorithm, assigning each BMM component to its nearest neighbor is quite different step from the algorithms in [62, 63, 191, 175, 189] where Euclidean or Hamming distance can simply be used. In our case, the distance (similarity) between two Bernoulli components should be calculated. The most straightforward way to do this is to use the Kullback-Leibler (KL) divergence. For discrete probability distributions f and g, the KL divergence of g from f is defined as

$$D(f\|g) = \sum_x f(x) \log \frac{f(x)}{g(x)}. \tag{5.16}$$

From this definition, the KL divergence between two *univariate* Bernoulli distributions $f(x) = \mu_1^x(1-\mu_1)^{1-x}$ and $g(x) = \mu_2^x(1-\mu_2)^{1-x}$ is obtained as

$$D(f\|g) = \mu_1 \log \frac{\mu_1}{\mu_2} + (1-\mu_1) \log \frac{(1-\mu_1)}{(1-\mu_2)}. \tag{5.17}$$

Because the KL divergence is additive for independent distributions, the KL divergence between two *multivariate* Bernoulli distributions $p_i(x)$ and $p_j(x)$ is obtained as

$$D(p_i\|p_j) = \sum_{d=1}^{D} \left(\mu_{id} \log \frac{\mu_{id}}{\mu_{jd}} + (1-\mu_{id}) \log \frac{(1-\mu_{id})}{(1-\mu_{jd})} \right). \tag{5.18}$$

In this thesis, $D(p_i\|p_j)$ is used in clustering BMM components.

Figure 5.1 illustrates the data structure of a randomized BMM tree. The root node only has the pointers to its child nodes. The inner node has a BMM component with

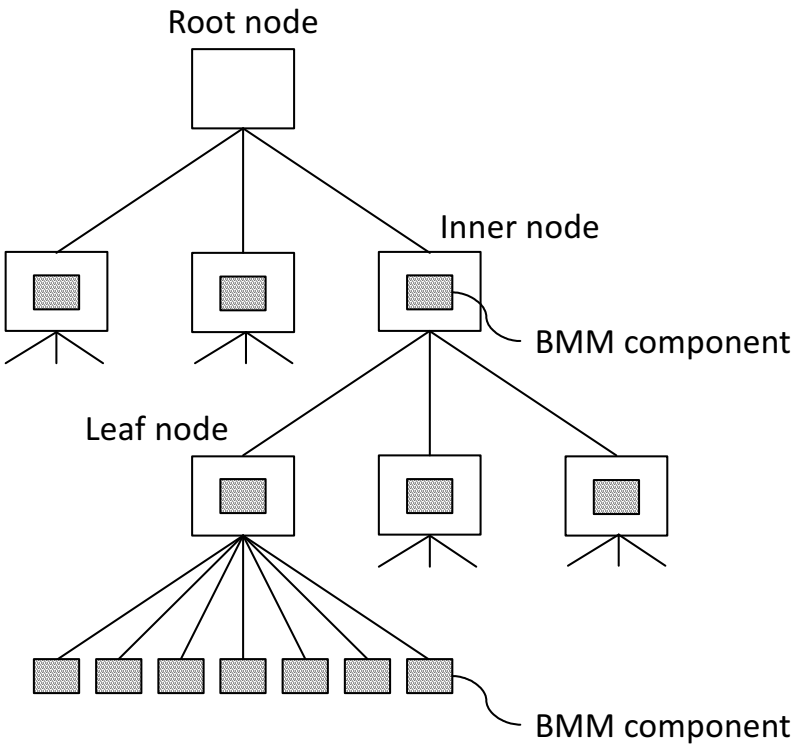


Figure 5.1: Structure of a randomized BMM tree.

parameters $\{\mu_{id}\}_{d=1}^{D}$ and the pointers to its child nodes. The leaf node has a BMM component and the pointers to the BMM components.

Searching in Randomized BMM Trees

Algorithm 3 shows the algorithm to search randomized BMM trees. In this algorithm, multiple randomized BMM trees are searched simultaneously using a single priority queue. It starts with recursively traversing each of trees in a depth-first manner (Line 4-7). While this traversals, each of the unexplored nodes is added to the priority queue PQ according to log-likelihood $\log p_i(x)$, where i denotes the identifier of the BMM component assigned to the node (Line 21). When the traversal reaches the leaf node, $\log p_i(x)$ is calculated against all BMM components assigned to the leaf node. After all the trees have been searched once, the search is continued by picking up the top node from the priority queue PQ and performing the tree traversal starting from the node. The search stops when the predefined number L of BMM components are searched.

Algorithm 3 Searching randomized BMM trees

Input: randomized BMM treesF T_i, query binary feature x
Output: A BMM components with largest posterior probability

$L \leftarrow 0$ ▷ number of points searched
$PQ \leftarrow \emptyset$ ▷ priority queue for unexplored nodes
$R \leftarrow \emptyset$ ▷ priority queue for temporal results
for each tree T_i **do**
    $N_i \leftarrow$ the root node of T_i
    TraverseTree(N_i, PQ, R)
end for
while $PQ \neq \emptyset \wedge L < L_{\max}$ **do**
    $N \leftarrow$ top node of PQ
    TraverseTree(N, PQ, R)
end while
return top-A BMM components from R
procedure TraverseTree(N, PQ, R)
    if node N is a leaf node **then**
        search all BMM components in N and add them to R
        $L \leftarrow L + |N|$
    else
        $C \leftarrow$ childnodesof$N$
        $C_q \leftarrow$ closestnodeofCto query$x$
        $C_p \leftarrow C \setminus C_q$
        add all nodes in C_p to PQ
        TraverseTree(C_q, PQ, R)
    end if
end procedure

5.1.6 Fast Posterior Calculation with SIMD Operations

Though the randomized BMM trees do accelerate finding the component with maximum posterior probability $\arg\max p(i|x)$, there is room for improvement in posterior calculation itself, which is required in the traversal of the trees. More precisely, it is required to calculate log-likelihood $\log p_i(x)$:

$$\log p_i(x) = \sum_{d=1}^{D} \left(x_d \log \mu_{id} + (1 - x_d) \log(1 - \mu_{id}) \right). \tag{5.19}$$

It is easy to calculate $\log p_i(x)$; for each bit x_d, $\log \mu_{id}$ is added if $x_d = 1$, otherwise $\log(1 - \mu_{id})$ is added. However, it involves D iterations, and thus is not efficient. One approach to accelerate this is the use of lookup table. For example, for each byte of x, all possible values (256 patterns) of corresponding sum of log-likelihood are pre-computed and stored in lookup table. Once x is given, we can get log-likelihood by simply looking up the table for each byte. However, this approach requires a large amount of memory and its performance is not as good as expected. Therefore, we accelerate this calculation using a trick.

Let $\log \overline{\mu_{id}}$ denote $\log(1 - \mu_{id})$. In indexing, we store $r_d = \log \mu_{id} - \log \overline{\mu_{id}}$ instead of storing both $\log \mu_{id}$ and $\log \overline{\mu_{id}}$. In querying time, we first sum up $r = (r_1, \cdots, r_d, \cdots, r_D)$ over all d such that $x_d = 1$. This is efficiently done by masking $r \wedge x$. Then, $\sum_{d=1}^{D} \log \overline{\mu_{id}}$ is added, which can be pre-computed and stored in database. While this gives us the same result as Eq. (5.19), the above AND and SUM operations can be accelerated using SIMD operations.

Figure 5.2 illustrates this tricky calculation of $\log p_i(x)$ in case of $D = 4$. Firstly, the vector of $r = (r_1, \cdots, r_4)$ is masked using x, removing r_d s.t. $x_d = 0$. Secondly, the masked r is summed up as $r^\top \mathbf{1} = r_2 + r_3 = \log \mu_{i2} - \log \overline{\mu_{i2}} + \log \mu_{i3} - \log \overline{\mu_{i3}}$. Finally, $\sum_{d=1}^{D} \log \overline{\mu_{id}}$ is added, resulting in $\log \overline{\mu_{i1}} + \log \mu_{i2} + \log \mu_{i3} + \log \overline{\mu_{i4}}$.

In our implementation, we store scalar-quantized r_d as follows. In order to avoid $\log 0$, μ_{id} is bounded bounded in the range $0.01 \leq \mu_{id} \leq 0.99$ in parameter estimation. Therefore, assuming that the base of logarithms is 10, the inequality $-2 < r_d < 2$ is satisfied. Scaled by a factor of 50, r_d is stored as 1-byte signed char. Using 128-bit SIMD operations, 16 variables can be processed simultaneously in the AND and SUM operations.

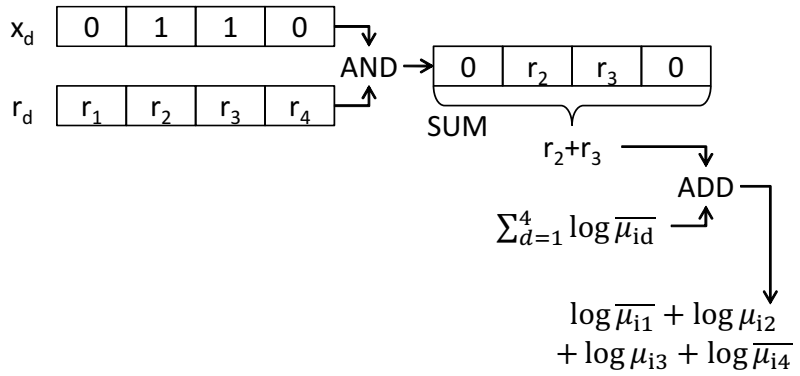


Figure 5.2: A tricky calculation of $\log p_i(x)$ using SIMD operations.

5.2 Experimental Evaluation

In the experiments, the Stanford mobile visual search dataset[1] is used. It contains eight classes of images: camera-phone images of books, business cards, CDs, DVDs, outdoor landmarks, museum paintings, text documents, and video clips. Each class consists of 100 reference images and 400 query images. As an indicator of retrieval performance, mean average precision (MAP; higher is better) [77] is used. We adopt the ORB feature [19] implemented in OpenCV library[2], where at most 900 features are extracted from 4 scales on average. The parameters of BMM are trained using the MIR Flickr collection[3].

5.2.1 Evaluation of Searching with Randomized BMM Trees

We first evaluate the search performance with randomized BMM trees proposed in Section 5.1.5. We learnt the BMM parameters for $N = 1,000$ and $N = 20,000$ using training binary vectors extracted from images in the MIR Flickr collection. Then, 300,000 binary vectors distinct from the above training vectors are used as query. Search precision is defined as the probability that the BMM component with the largest posterior $p(i|x)$ is found. Of course, linear search always finds it.

Figure 5.3 shows the evaluation results of approximate nearest neighbor search using randomized BMM trees. Search time per query point as a function of search precision is shown for 1, 2, 4, 8, 16 trees with $L_{\max} = 32, 64, 128, 256$, where $L_{\max}$ is the number of points to be searched. The branching factor K and the maximum

[1]https://sites.google.com/site/chenmodavid/mobile-visual-search
[2]http://opencv.org/
[3]http://press.liacs.nl/mirflickr/

leaf size S_L are fixed as $K = 32$ and $S_L = 100$. The SIMD optimization introduced in Section 5.1.6 is used here. In case of linear search, we require 4 [ms] for $N = 1,000$ and 84 [ms] for $N = 20,000$.

We can see that randomized BMM trees do accelerate the calculation of $\arg\max_i p(i|x)$. For $N = 1,000$, a single tree seems to be sufficient; the overhead of using multiple trees becomes relatively large for small NNS problem. On the other hand, for $N = 20,000$, using multiple trees is better choice if high precision is required; it is better to search multiple trees instead of recursively searching a single tree in large NNS problem. Searching in one tree with $L = 256$ for $N = 1,000$ is 14x faster than linerar search, achieving the search precision of 93.9%. Searching in two trees with $L = 256$ for $N = 20,000$ is 132x faster than linerar search, achieving the search precision of 65.1%.

Figure 5.4 shows search time per query point as a function of search precision for the different methods in calculating log-likelihood $\log p_i(x)$. The number of trees is fixed to 4. SIMD is the SIMD optimized search method introduced in Section 5.1.6. This is the same as 4trees in Figure 5.3. LUT is the search method using lookup table. NAIVE is the search method naively calculate $\log p_i(x)$. It is found that LUT achieves somewhat disappointing result; there is not so much improvement from naive implementation. This is because CPU cache does not work effectively; lookup tables requires a large amount of memory, and these lookup tables are used only once, resulting in cache miss in CPU. Comparing SIMD and NAIVE, SIMD is about 3x faster than the naive implementation for $N = 1,000$ and 3x to 5x faster for $N = 20,000$. The difference between SIMD and NAIVE becomes larger for larger N, smaller number of trees, and larger number of L, where the computational cost for the calculation of $\log p_i(x)$ is dominant compared with the overhead of the tree traversal.

While the use of lookup tables does not degrades the search precision, the SIMD operation does because it involves quantization of the parameters as described in Section 5.1.6. The maximum degradation was 3.1% for 16 trees, $N = 20,000$, and $L = 256$, achieving 3.5x faster search than naive implementation. We think the degradation of search precision is negligible in compensation for such acceleration.

Table 5.5 compares maximum posterior probabilities $\arg\max_i p(i|x)$ obtained by the exact linear search, randomized BMM trees, and randomized BMM trees with quantization of parameters (SIMD). In Table 5.5 (a), we can see that obtain accurate $\max_i p(i|x)$ is obtained with the randomized BMM trees. The points not at the line $y = x$ correspond to the cases that wrong $\arg\max_i p(i|x)$ is obtained with the randomized BMM trees. In Table 5.5 (b), it is shown that $\max_i p(i|x)$ is distorted by

the randomized BMM trees with quantization. This is an implementation issue. We quantized $r_d = \log \mu_{id} - \log \overline{\mu_{id}}$ into uchar value using a floor function. The distortion would be solved if this floor function is replaced by a round-off function.

5.2.2 Evaluation of BMM-VW

We first evaluate the simplest proposed method, BMM-VW. Table 5.1 compares the accuracy (mean average precision) of BoBW [17] and BMM-VW methods. The number of VWs in BoBW and the number of mixture components in BMM-VW is fixed to 20,000. For BoVW, linear search is performed in quantization (Linear). For BMM-VW, in addition to linear search, tree-based quantization proposed in Section 5.1.5 is performed, where four trees are used because linear search is not applicable in real use cases due to its high computational cost. The parameters of these trees are as follows: the branching factor $K = 32$, the maximum leaf size $S_L = 100$, and the maximum number of points to be searched $L_{\max} = 256$ and $L_{\max} = 32$. This corresponds to the left end ($L_{\max} = 32$) and the right end ($L_{\max} = 256$) of 4 trees in Figure 5.3 (b).

In Table 5.1, we can see that BMM-VW outperforms BoBW on all classes. This is because BMM can capture complex bit dependencies that underlie binary features compared with BoBW as shown in Chapter 3. Comparing BMM-VW Linear and BMM-VW $L_{\max} = 256$, there is no large degradation in accuracy if tree-based approximate quantization is performed, while BMM-VW $L_{\max} = 256$ is 102x faster than BMM-VW Linear. However, comparing BMM-VW $L_{\max} = 256$ and BMM-VW $L_{\max} = 32$, MAP declines to the same level as BoBW, where precision of tree-based search declines from 74.4% ($L_{\max} = 256$) to 32.7% ($L_{\max} = 32$). The accuracy of tree-based search does not have to be perfect. This is because two features extracted reference and query image can be matched if only they are quantized into the same Bernoulli component; they are not necessarily be quantized into the Bernoulli component with maximum (log-)likelihood.

5.2.3 Evaluation of BMM-FK

The method BMM-FK proposed in Section 5.1.4 is evaluated. Here, the number of BMM components is fixed to 1,000. In quantization, the randomized BMM trees proposed in Section 5.1.5 is adopted for acceleration, where a single tree is used with the following parameters: the branching factor $K = 32$, the maximum leaf

Table 5.1: Comparison of BoBW and BMM-VW on eight classes in terms of mean average precision.

	BoBW	BMM-VW	BMM-VW	BMM-VW
Quantization type	Linear	Linear	Tree $L_{\max} = 256$	Tree $L_{\max} = 32$
book	0.689	0.714	0.713	0.696
cards	0.174	0.183	0.183	0.175
cd	0.577	0.611	0.613	0.578
dvd	0.611	0.642	0.636	0.611
landmarks	0.122	0.133	0.127	0.098
paintings	0.576	0.587	0.585	0.581
text	0.205	0.224	0.225	0.219
video	0.690	0.693	0.688	0.659
average	0.455	0.474	0.471	0.452

size $S_L = 100$, and the maximum number of points to be searched $L_{\max} = 256$. This corresponds to the right end ($L_{\max} = 256$) of 1 tree in Figure 5.3 (a).

In this section, three types of distance/similarity are compared: Hamming, FK, and FV ℓ_2. Hamming is a standard Hamming distance used in Chapter 4. FK is the similarity value of the Fisher kernel defined in Eq. (5.12). FV ℓ_2 is the distance between Fisher vectors of binary features. This is derived by the explicit feature mapping of the Fisher kernel in Eq. (5.12) as:

$$\alpha_i(x)^\top F_i^{-1} \alpha_i(y) = \left(F_i^{-1/2} \alpha_i(x)\right)^\top \left(F_i^{-1/2} \alpha_i(y)\right). \tag{5.20}$$

Letting $\mathbf{v}_i(x)$, the Fisher vector representation of x, denote $F_i^{-1/2}\alpha_i(x)$, FV ℓ_2 between x and y is defined as

$$d_{\mathrm{FV}\ell_2}(x, y) = \|\mathbf{v}_i(x) - \mathbf{v}_i(y)\|_2, \tag{5.21}$$

where $i = \arg\max_j p(j|x) = \arg\max_j p(j|y)$.

As a scoring fuction, three types of scoring methods are used: Const, LN(b), and FK. Const is to vote a constant value (e.g. 1.0) as a score. LN(b) is a scorng method proposed in Section 4.2.5. As this scorig function requires the distances between a query feature and reference features, it can be combined with only Hamming and FV ℓ_2. FK votes the value of the Fisher kernel in Eq. (5.12) as a score.

Table 5.2 shows the comparison of the combination of distances and scoring functions. In this experiment, 256-bit binary feature vectors are stored in the inverted inedx to calculate distance or similarity between a query feature and a reference feature. We can see that FK achieves the best performance if the constant score is

used, which indicates that FK is more appropriate similarity measurement than the Hamming distance and the ℓ_2 distance of the Fisher vector. However, if non-constant scores are used, the combination of FV ℓ_2 dissimilarity measurement and the LN(b) scoring achieved the best performance. This implies that the *order* of FK similarity is very usuful, but the values of the similarity are not appropriate to being directly used as scores.

Bit Selection, Multiple Assignment, and Weak Geometric Consistency

Next, we apply the bit selection method explained in Chapter 4 to these methods in order to reduce the memory consumption of the inverted index. Table 5.3 compares the combination of distances and scoring functions, where 64-bit substrings are used in the calculation of distance or similarity instead of full 256-bit descriptors. It is found that the degradation of accuracy from full 256-bit descriptors is larger than the case of the extended inverted index method explained in Chapter 4. This is because, in BMM-FK, the distribution of binary features is appropriately modeled and reflected to the scoring function, while the extended inverted index method does not consider the underlying distribution. Some bits whose mean values are far from 0.5 become *noise* to the extended inverted index method. The bit selection method can remove these bits. Therefore, the bit selection does not degrade the accuracy of the extended inverted index method or even improve it as shown in Chapter 4. On the other hand, BMM-FK makes the best of the information of all bits even if the mean values of bits are far from 0.5. Therefore, the bit selection has larger impact on BMM-FK .

Table 5.4 shows the comparison of the combination of distances and scoring functions when the multiple assignment (MA) is adopted in order to improve the recall of the feature-level matching. As expected, the accuracy is improved in all methods. While the best performance is achieved by the FK similarity measure for constant scoring, the combination of the Hamming distance and the LN(b) scoring achieved the best performance for non-constant scoring. Table 5.5 shows the omparison of the combination of distances and scoring functions when 64-bit substrings and MA are used. Similarly to Table 5.4, the best performance is achieved by the combination of the Hamming distance and the LN(b) scoring.

Table 5.6 shows the comparison of the combination of distances and scoring functions on 256-bit binary feature vectors when MA and weak geometric consistency

(WGC)[4] are adopted. Compared with Table 5.4 (w/o WGC), the accuracy is improved for all methods. The combination of the Hamming distance and the the LN(b) scoring achieves the highest MAP 0.814.

Normalized FV ℓ_2

It is known that ℓ_2 normalization can boost the performance of the Fisher vector [116] or the VLAD representation [129]. In Chapter 3, it is also shown that the proposed Fisher vector of BMM is improved by ℓ_2 normalization. Here, we evaluate a normalized FV ℓ_2 distance between x and y that is defined as:

$$\begin{aligned} d_{\mathrm{NFV}\ell_2}(x,y) &= \left\| \frac{\mathbf{v}_i(x)}{\|\mathbf{v}_i(x)\|_2} - \frac{\mathbf{v}_i(y)}{\|\mathbf{v}_i(y)\|_2} \right\|_2 \\ &= 2 - \frac{\mathbf{v}_i(x)^\top \mathbf{v}_i(y)}{\|\mathbf{v}_i(x)\|_2 \|\mathbf{v}_i(y)\|_2}. \end{aligned} \tag{5.22}$$

Table 5.7 evaluates the normalized FV ℓ_2 (NFV ℓ_2) distance, where 256-bit binary feature vectors are used in the calculation of the distance. For scoring, in addition to Const and LN(b), gaussian weighting (GW) is also compared. For GW, the optimized parameter $\sigma = 0.5$ is used as shown in Figure 5.6. From Table 5.7, we can see that the NFV ℓ_2 distance with the LN(b) scoring achieves the highest accuracy compared with the other combinations distance/similarity and scoring methods. In Table 5.8, we evaluate the NFV ℓ_2/LN(b) method combined with MA, WGC, and feature selection (FS) [182] as done in . We can see that MA and WGC further improve the accuracy of NFV ℓ_2/LN(b). The NFV ℓ_2/LN(b) method with MA and WGC outperforms the Hamming distance-based method in Table 5.6 (0.821 (NFV) vs 0.814(Hamming)). Furthermore, MAP is improved to 0.881 if feature selection is adopted. This significantly outperforms the system in Chapter 4 (MAP 0.811). Table 5.9 shows the performance evaluation of the normalized FV with 64-bit substrings. Although the accuracy of the normalized FV with 64-bit declined from one with 256-bit full string, it is still better than the system in Chapter 4.

[4]We used WGC only for the verification of orientation consistency because scale consistency is not effective compared with orientation.

Table 5.2: Comparison of the combination of distances and scoring functions. 256-bit binary feature vectors are used in the calculation of distance or similarity.

Distance/Similarity	Hamming		FV ℓ_2		FK	
Scoring	Const	LN(b)	Const	LN(b)	Const	FK
book	0.928	0.962	0.929	0.960	0.941	0.949
cards	0.589	0.693	0.603	0.700	0.611	0.659
cd	0.823	0.889	0.826	0.898	0.848	0.868
dvd	0.896	0.962	0.913	0.964	0.925	0.941
landmarks	0.258	0.326	0.265	0.326	0.265	0.298
paintings	0.618	0.780	0.645	0.780	0.660	0.700
text	0.569	0.660	0.582	0.674	0.590	0.628
video	0.815	0.907	0.831	0.909	0.876	0.888
average	0.687	0.772	0.699	0.776	0.715	0.741

Table 5.3: Comparison of the combination of distances and scoring functions. 64-bit substrings are used in the calculation of distance or similarity.

Distance/Similarity	Hamming		FV ℓ_2		FK	
Scoring	Const	LN(b)	Const	LN(b)	Const	FK
book	0.912	0.953	0.926	0.951	0.926	0.945
cards	0.547	0.583	0.551	0.567	0.570	0.609
cd	0.787	0.862	0.808	0.845	0.823	0.843
dvd	0.902	0.950	0.894	0.939	0.910	0.921
landmarks	0.204	0.222	0.196	0.210	0.202	0.215
paintings	0.611	0.738	0.619	0.730	0.639	0.685
text	0.539	0.600	0.540	0.593	0.560	0.570
video	0.825	0.894	0.837	0.885	0.861	0.881
average	0.666	0.725	0.671	0.715	0.686	0.708

5.3 Summary

In this Chapter, starting with general match kernel, it is proposed to integrate the essence of the Fisher kernel and the BoVW framework. Firstly, similarity between binary features is defined by the Fisher kernel. Using the assumption that posterior probability is peaky, the Fisher match kernel is linked with the BoVW framework, resulting in two proposed method: BMM-VW and BMM-FK. BMM-VW can be considered as a variant of BoVW, where VWs are defined by the BMM components instead of representative binary features. BMM-FK is a BoVW-based framework very similar to one proposed in Chapter 4. The differences are (1) VWs are defined by BMM, and (2) similarity of binary features is defined by the Fisher kernel. In this chapter, the method called randomized BMM trees is also proposed, which

Table 5.4: Comparison of the combination of distances and scoring functions. 256-bit binary feature vectors are used in the calculation of distance or similarity. Multiple assignmet is adopted.

Distance/Similarity	Hamming		FV ℓ_2		FK	
Scoring	Const	LN(b)	Const	LN(b)	Const	FK
book	0.947	0.966	0.940	0.961	0.957	0.957
cards	0.660	0.740	0.616	0.705	0.677	0.682
cd	0.865	0.908	0.840	0.893	0.870	0.870
dvd	0.948	0.982	0.932	0.971	0.962	0.958
landmarks	0.327	0.380	0.275	0.324	0.291	0.301
paintings	0.643	0.812	0.638	0.796	0.701	0.732
text	0.637	0.708	0.590	0.654	0.658	0.667
video	0.854	0.927	0.836	0.904	0.897	0.912
average	0.735	0.803	0.708	0.776	0.752	0.760

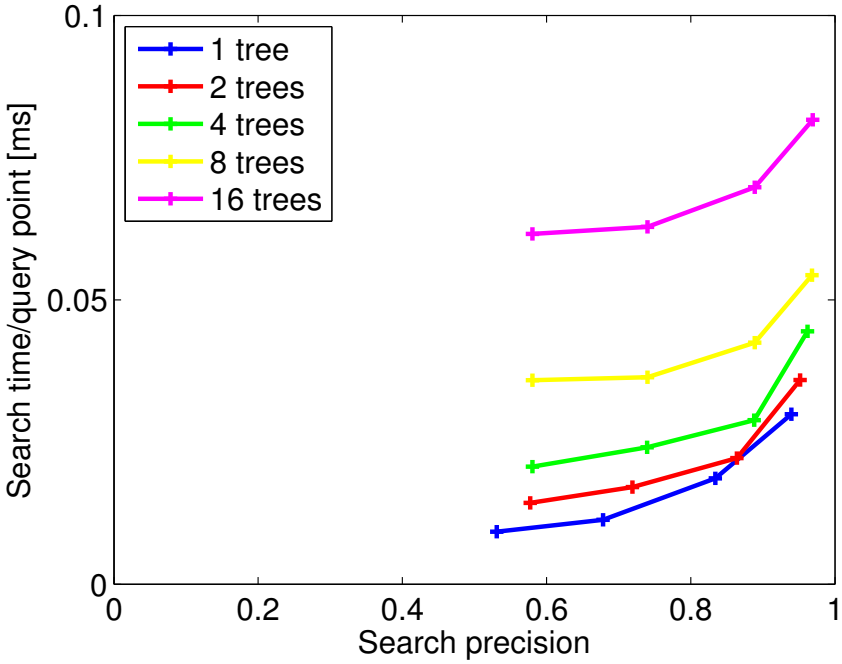
significantly accelerates the calculation of the quantization in BMM-VW and BMM-FK. Experimental results show that BMM-VW outperforms the BoBW representation. The evaluation of BMM-FK indicates that FK-based similarity measurement is very effective in terms of its *order*. However, it is found that using its value directly as a voting score is not appropriate. We believe that there is a room for improvement in exploiting the FK value in voting, and this is left as future work. On the other hand, the combination of the normalized Fisher vector distance and the modified version of the local NBNN scoring method in Chapter 4 achieved the best accuracy. By combining the normalized Fisher vector with multiple assignment, weak geometric consistency, and feature selection, the proposed normalized Fisher vector approach significantly outperforms the system proposed in Chapter 4 and the conventional state-of-the-art system in terms of the image retrieval accuracy.

Table 5.5: Comparison of the combination of distances and scoring functions. 64-bit substrings are used in the calculation of distance or similarity. Multiple assignmet is adopted

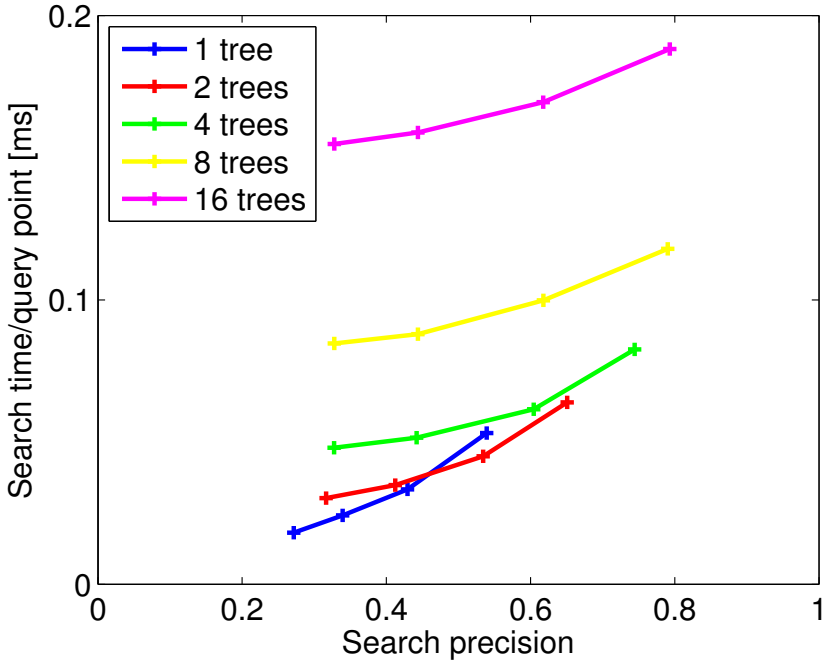
Distance/Similarity	Hamming		FV ℓ_2		FK	
Scoring	Const	LN(b)	Const	LN(b)	Const	FK
book	0.936	0.952	0.922	0.942	0.938	0.941
cards	0.595	0.631	0.560	0.577	0.581	0.592
cd	0.831	0.874	0.809	0.848	0.838	0.850
dvd	0.942	0.959	0.899	0.941	0.933	0.937
landmarks	0.219	0.249	0.191	0.207	0.209	0.221
paintings	0.665	0.766	0.621	0.721	0.707	0.722
text	0.584	0.629	0.536	0.580	0.564	0.575
video	0.844	0.885	0.823	0.863	0.882	0.887
average	0.702	0.743	0.670	0.710	0.706	0.716

Table 5.6: Comparison of the combination of distances and scoring functions. 256-bit binary feature vectors are used in the calculation of distance or similarity. Multiple assignmet and WGC is adopted.

Distance/Similarity	Hamming		FV ℓ_2		FK	
Scoring	Const	LN(b)	Const	LN(b)	Const	FK
book	0.958	0.972	0.948	0.968	0.963	0.964
cards	0.724	0.764	0.664	0.729	0.720	0.720
cd	0.892	0.919	0.883	0.909	0.895	0.898
dvd	0.973	0.985	0.970	0.975	0.973	0.971
landmarks	0.332	0.381	0.280	0.322	0.298	0.309
paintings	0.676	0.819	0.676	0.811	0.725	0.763
text	0.710	0.737	0.665	0.706	0.692	0.711
video	0.922	0.938	0.899	0.916	0.923	0.932
average	0.774	0.814	0.748	0.792	0.774	0.784

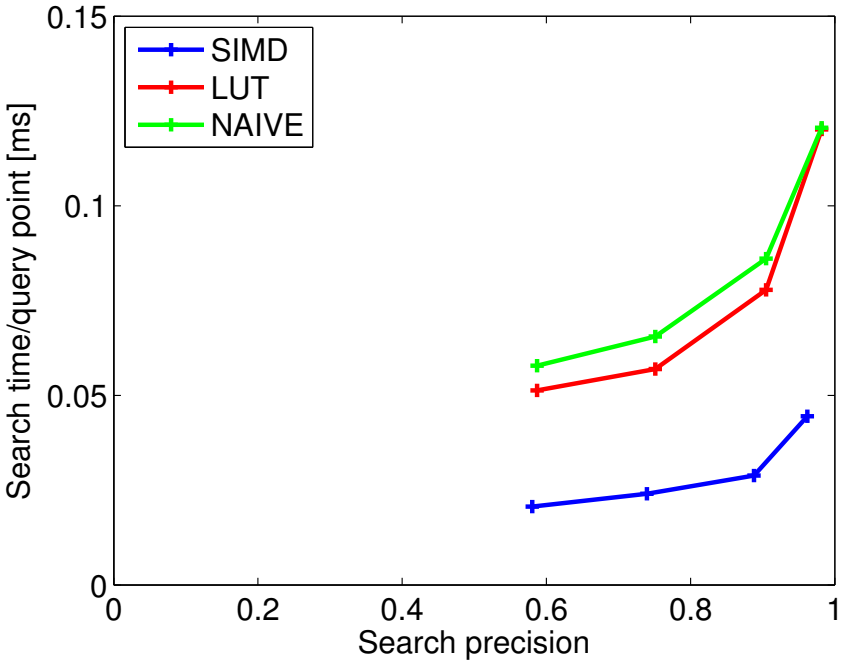


(a) $N = 1{,}000$.

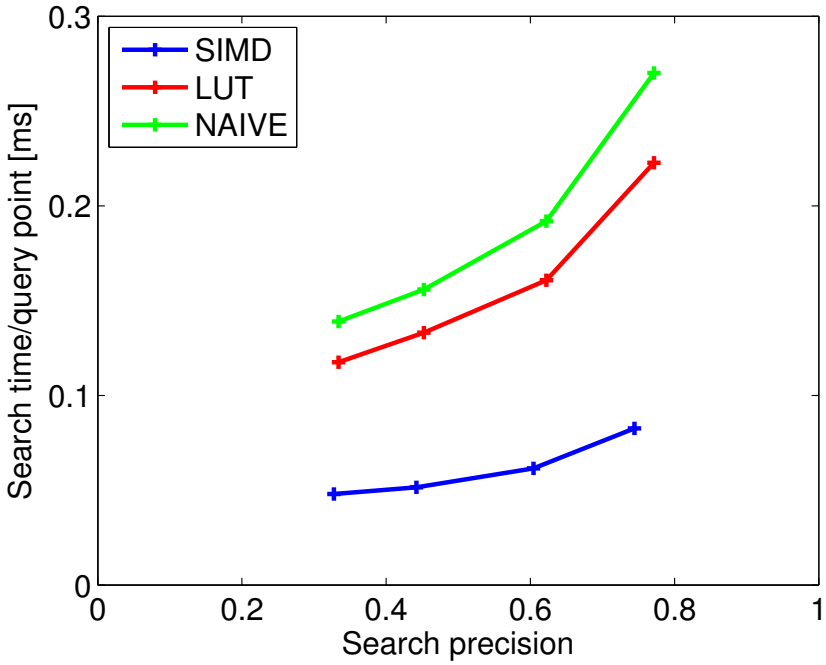


(b) $N = 20,000$.

Figure 5.3: Search time per query point as a function of search precision for 1, 2, 4, 8, 16 trees with $L = 32, 64, 128, 256$.

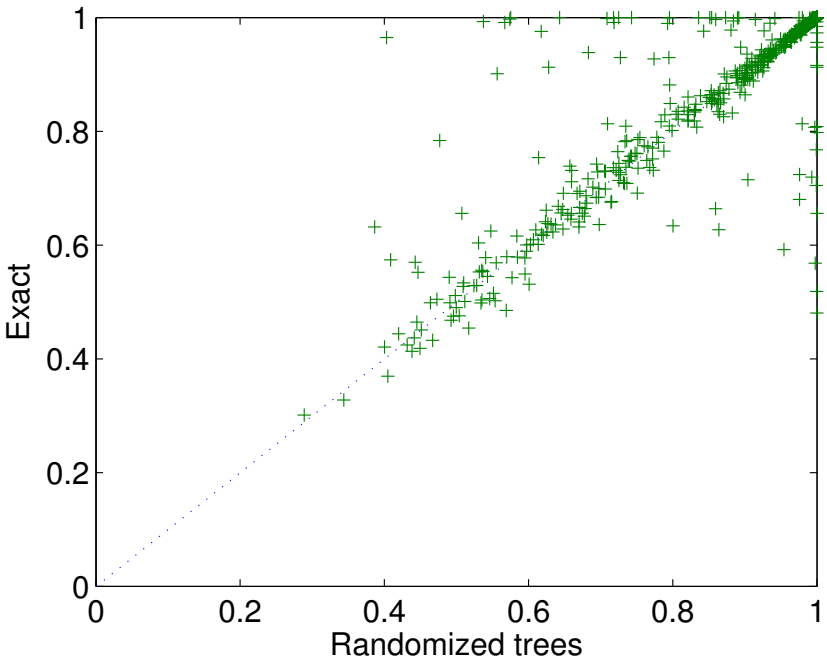


(a) $N = 1{,}000$.

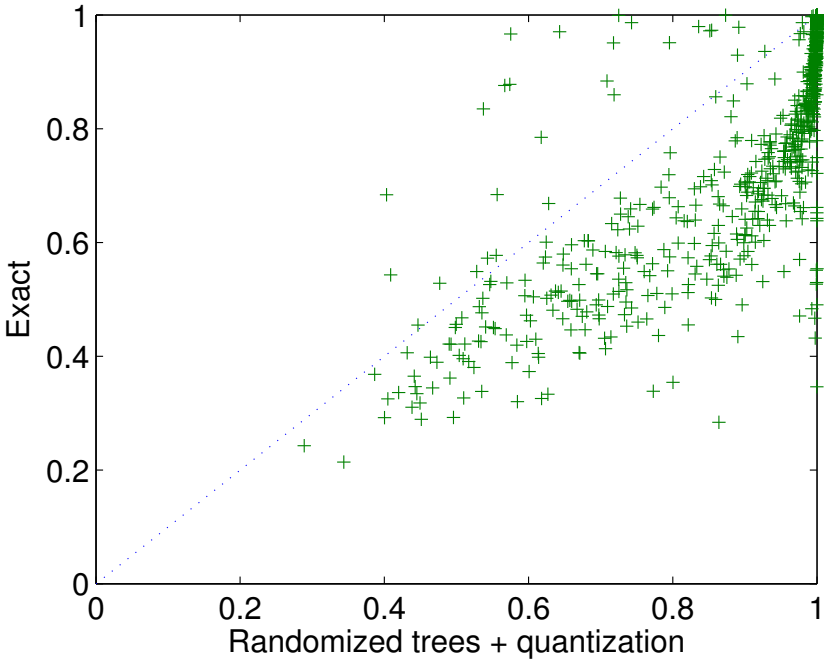


(b) $N = 20,000$.

Figure 5.4: Search time per query point as a function of search precision for the different methods in calculating log-likelihood $\log p_i(x)$ with $L = 32, 64, 128, 256$. The number of trees is fixed to 4.



(a) exact linear search (y-axis) vs randomized BMM trees (x-axis)



(b) exact linear search (y-axis) vs randomized BMM trees with quantization (x-axis)

Figure 5.5: A comparison of maximum posterior probabilities $\arg\max_i p(i|x)$ obtained by the exact linear search, randomized BMM trees, and randomized BMM trees with quantization of parameters (SIMD). 900 binary features from an image are plotted for this evaluation.

Table 5.7: Evaluation of the normalized FV ℓ_2 distance. 256-bit binary feature vectors are used in the calculation of the distance.

Distance	NFV ℓ_2		
Scoring	Const	LN(b)	GW
book	0.935	0.966	0.964
cards	0.623	0.716	0.703
cd	0.835	0.897	0.898
dvd	0.925	0.968	0.960
landmarks	0.276	0.340	0.342
paintings	0.653	0.774	0.757
text	0.595	0.684	0.631
video	0.857	0.916	0.917
average	0.712	0.782	0.771

Table 5.8: Comparison of the combination of distances and scoring functions. 256-bit binary feature vectors are used in the calculation of distance or similarity. Multiple assignmet, WGC, and feature selection are adopted.

Distance/Scoring	NFV ℓ_2/LN(b)				
	MA1	MA3	MA5	+WGC	+FS
book	0.966	0.966	0.966	0.970	0.984
cards	0.716	0.750	0.752	0.783	0.888
cd	0.897	0.912	0.913	0.920	0.963
dvd	0.968	0.980	0.982	0.984	0.989
landmarks	0.340	0.373	0.388	0.409	0.504
paintings	0.774	0.806	0.808	0.807	0.842
text	0.684	0.720	0.722	0.748	0.884
video	0.916	0.939	0.930	0.950	0.992
average	0.782	0.806	0.808	0.821	0.881

Table 5.9: Comparison of the combination of distances and scoring functions. 64-bit binary feature vectors are used in the calculation of distance or similarity. Multiple assignmet, WGC, and feature selection are adopted.

Distance/Scoring	NFV ℓ_2/LN(b)				
	MA1	MA3	MA5	+WGC	+FS
book	0.955	0.960	0.961	0.958	0.981
cards	0.621	0.665	0.659	0.691	0.834
cd	0.866	0.879	0.884	0.887	0.954
dvd	0.953	0.966	0.967	0.975	0.982
landmarks	0.264	0.281	0.286	0.276	0.380
paintings	0.726	0.746	0.742	0.767	0.793
text	0.616	0.641	0.654	0.680	0.844
video	0.893	0.902	0.899	0.934	0.990
average	0.737	0.755	0.756	0.771	0.845

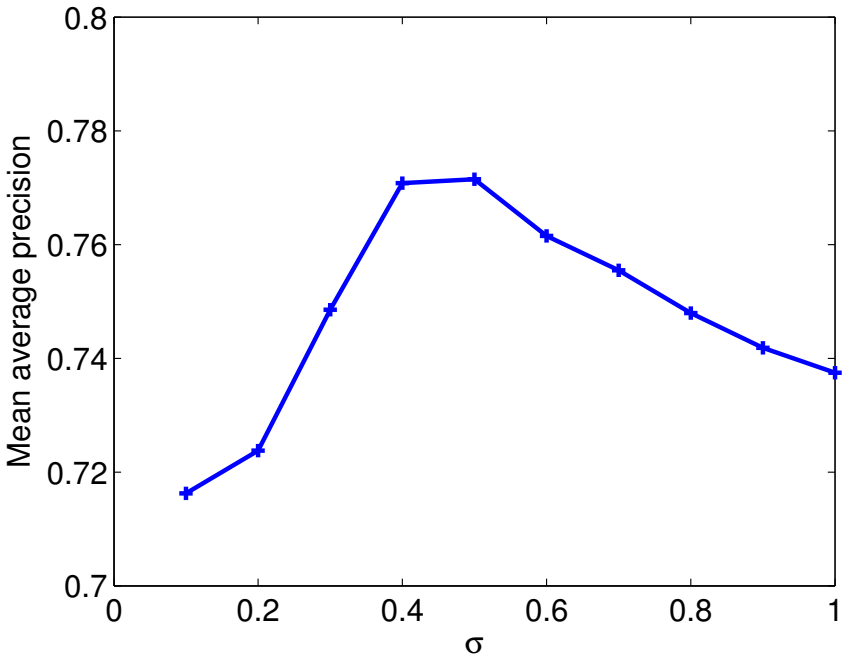


Figure 5.6: MAP averaged over all classes for the normalized FV ℓ_2 with gaussian weighting as a function of the parameter σ. The highest accuracy 0.771 is achieved at $\sigma = 0.5$.

Chapter 6

Applications

In this chapter, I introduce the real applications where a part of this work has been used as a image retrieval engine. Especially, the method proposed in Chapter 4 is used as a core engine in the following applications and systems.

6.1 SATCH VIEWER

SATCH VIEWER[1] is an AR application provided by KDDI Corp.[2], whicn recognizes various items surrounding us such as posters, catalogs, leaflets, boxes of sweets, and so on. Once an object in the database is recognized, AR content is downloaded and shown to the application user. The AR content can easily be customized for each object, there are many use cases other than showing standard AR content; showing movies, providing point service or stamp rally service, linking to social networking services, and so on. Figure 6.1 illustrates the screenshots of the application. In this application, a server-client system is used because the database of recognizable items frequently changes.

6.2 Catalog Camera

Catalog camera is a mobile application which had been provided by Nissen Co., Ltd[3], which can *augment* clothing catalogs. If you are interested in a dress on the catalog, you can get more information unavailable from the catalog by just

[1] http://viewer.satch.jp/
[2] http://www.kddi.com/corporate/news_release/2013/0117a/index.html
[3] http://www.nissen-hd.co.jp/ir/pdf/IR_13_01_17_2.pdf

(a) query image. (b) recognition result with AR content.

Figure 6.1: Screenshots of SATCH VIEWER application.

capturing the catalog with your smartphone. Figure 6.2 shows a screenshot of the application. The additional information includes user reviews of clothing, special discount information, direct link to buy clothing, and so on.

This application is first implemented as a server-client system. However, as each of catalogs consists of only a few hundred pages and its content does not change, it is replaced by a stand-alone image recognition system, where the database of catalog pages are included in the application itself. The small-memory-footprint characteristics of the method proposed in Chapter 4 had enabled this application.

6.3 au PLAY SCREEN

The third application is au PLAY SCREEN campaign by KDDI Corp. In this campaign, ODOROKI application had been provided, which recognizes posters, movies, banner advertisements, and TV commercials which are related to au PLAY SCREEN campaign. Once one of the above content is recognized, a drama story proceeds according to the recognized content in the smartphone. Figure 6.3 shows a sample image of this campaign[4], where a campaign poster is to be recognized.

[4]http://www.au.kddi.com/odoroki/

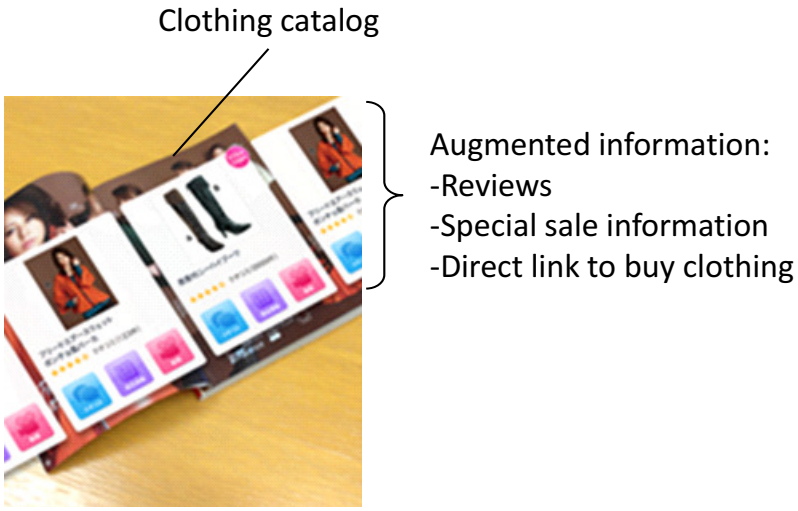


Figure 6.2: A screenshot of Catalog Camera application showing AR content related to the catalog.

In this application, a hybrid system of a server-client system and a stand-alone system is adopted[5]. In this hybrid system, a captured image is first recognized by a stand-alone system. If there is no result obtained in local search, the application sends the captured image to the recognition server. The advantage of this hybrid system is load reduction of the recogntion server; if the search is completed inside the local system, no query is send to the server. This is especially effective against queries related to TV commercials because these queries tend to occur simultaneously when the TV commercials come on. In indexing movies, we developed an algorithm to reduce the number of frames. Firstly, coarsely sampled frames are indexed. Then, additional frames are gradually added if the frame had not been recognized by the neighboring frames which are already indexed. This algorithm has reduced the number of indexed frames by 86.2%.

6.4 Kaimono Camera

Kaimono (meaning *shopping*) Camera is an application provided by Dentsu Inc.[6] This application recognizes more than one million products which can be found on the web sites of kakaku.com[7] and @cosme[8], including book covers, DVD/Blu-ray Discs,

[5]http://www.gizmodo.jp/2013/08/au_4glte_play_screen.html
[6]http://kaimonocamera.com/
[7]http://kakaku.com/
[8]http://www.cosme.net/

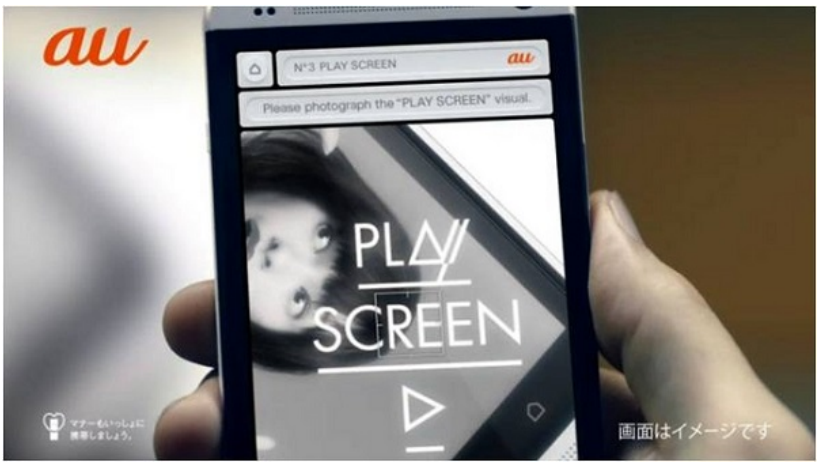
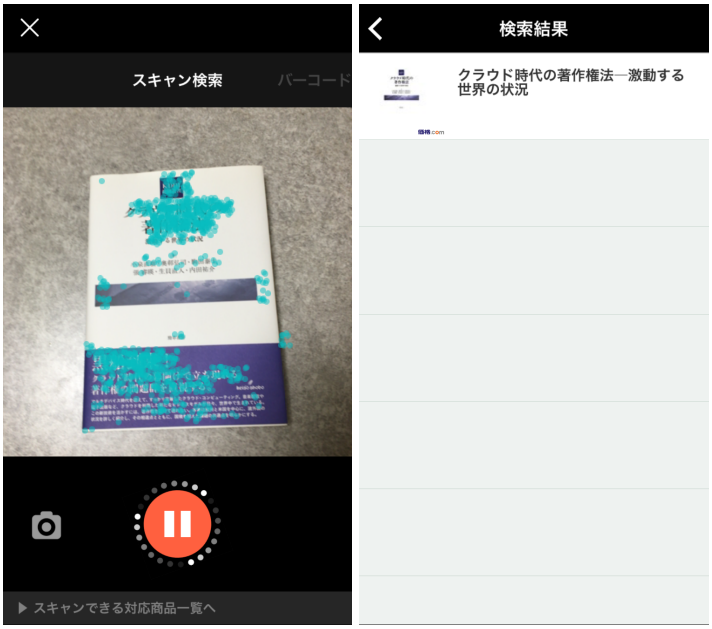


Figure 6.3: Sample image of au PLAY SCREEN campaign.

CDs, and bottles of liquor. Using this application, users can easily check or compare the prices of a product by capturing the product with a smartphone. Figure 6.4 shows screenshots of Kaimono Camera application. The application continuously captures images and recognizes them. Once a product is found in the database, the application screen immediately changes to the result screen. In this application, a server-client system is used to realize large-scale recognition.



(a) recognizing the book. (b) recognition result.

Figure 6.4: Screenshots of Kaimono Camera application.

Chapter 7

Conclusion

With the increasingly wide-spread use of mobile devices such as Android phones or iPhones, mobile visual search (MVS) has become one of the major applications of image retrieval and recognition technology. With MVS, we can recognize the surrounding world with mobile devices using its built-in camera as an input to image recognition or retrieval systems. Mobile devices can be also good interface to effectively show the retrieval or recognition results using augmented reality technology, for instance. Recent Head Mounted Display (HMD) might be also the good platform for image retrieval researches. Considering these trends, in this thesis, we developed a practical local MVS system using recent binary features by proposing the following three approaches.

- **Fisher vectors for binary features.** We proposed to apply the Fisher vector representation to binary features to improve the accuracy of binary feature-based image retrieval. Main contribution of this approach is to model binary features using the Bernoulli mixture model (BMM) and derive the closed-form approximation of the Fisher vector of BMM. To the best of my knowledge, this is the first time to model binary features with BMM and apply the Fisher vector approach. We showed that, by modeling binary features with BMM, it became possible to evaluate how informative different binary features are. Experimental results show that the proposed Fisher vector outperforms the BoVW method on various types of objects. In addition, we also propose a fast approximation method to accelerate the computation of the proposed Fisher vectors by one order of magnitude with comparable performance.

- **Extended inverted index for binary features.** We proposed a substring extraction method which extracts informative bits from original binary vector

and stores in the inverted index. These substrings are used to refine visual word-based matching. This was the first time to bring the idea of the Hamming embedding method to binary features. The advantage of this approach is its practicability. The developed system is very simple but effective, achieving good trade-offs between search precision, memory requirement, and speed. In addition, a modified version of the local naive Bayes nearest neighbor scoring method is proposed in the context of image retrieval, which considers the density of binary features in scoring each feature matching. Finally, in order to suppress false positives, we introduce a model check step after standard geometric verification using constraint on the configuration of a transformed reference image. The suppression of false positives is essential for real use cases. The proposed system could retrieve the database with one million images in 87 [ms] and its accuracy significantly outperformed that of the state-of-the-art local MVS system.

- **Linking Fisher kernel to inverted index-based systems.** We proposed to integrate the advantages of the above approaches. Starting with general match kernel, we show that the Fisher kernel-based similarity measurement can be implemented using the extended inverted index structure. Using the assumption that posterior probability is peaky, the Fisher kernel is linked with the BoVW framework, resulting in two proposed method, namely BMM-VW and BMM-FK. BMM-VW is a variant of BoVW, where VWs are defined by the BMM components. BMM-FK is the modified version of the second approach, where more appropriate similarity measurement is used. In order to ensure real-time applications, the method called randomized BMM trees is also proposed, which significantly accelerates the calculation of the quantization in BMM-VW and BMM-FK. In experiments, it was shown that the BMM-FK significantly outperformed the two previous approaches and the conventional state-of-the-art system in terms of the image retrieval accuracy.

Furthermore, we have developed real applications, which include a stand-alone system, a server-client system, and a hybrid system of them. Through these practical applications, it has been proven that our developed systems have sufficient potential for practical usages. Thus, in this thesis, we explored the potential of binary features in the area of image retrieval and established the basis of binary feature-based image retrieval that can be used to real applications.

Limitation & Future work

The most important limitation of our system is that we assumed the recognition targets are planar. In our system, we perform geometric verifications based on Homography matrix. While this process is very important to suppress false positives, it makes non-planar objects difficult to be recognized. Solving this problem has left for future work. One solution for this problem is to remove planar-assumption and use the Fundamental matrix in geometric verification. However, as the Degree of Freedom (DoF) of the Fundamental matrix is relatively large, it was difficult to achieve zero false positive in our preliminary experiments.

Recognition of deformable objects is also a problem. We face this problem in recognizing deformable product packages, for instance. Though there are several researches on deformable object recognition, the solutions tend to require high computational cost, which might not be acceptable for mobile devices.

In the third approach, it is shown that FK-based similarity measurement is very effective in terms of its *order*. However, it is found that using its value directly as a voting score is not appropriate. we believe that there is a room for improvement in exploiting the FK value in voting, and which is left as future work.

Bibliography

[1] Y. Rui, T. S. Huang, and S. Chang, "Image retrieval: Current techniques, promising directions and open issues," *JVCIR*, vol. 10, no. 1, pp. 39–62, 1999.

[2] H. Tamura and N. Yokoya, "Image database systems: A survey," *PR*, vol. 17, no. 1, pp. 29–43, 1984.

[3] V. N. Gudivada and J. V. Raghavan, "Special issue on content-based image retrieval systems," *IEEE Computer*, vol. 28, no. 9, pp. 18–22, 1995.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, H. Q, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content," *IEEE Computer*, vol. 28, no. 9, pp. 23–32, 1995.

[5] G. Pass and R. Zabih, "Histogram refinement for content-based image retrieval," in *Proc. of WACV*, 1996.

[6] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, "Image indexing using color correlograms," in *Proc. of CVPR*, 1997, pp. 762–768.

[7] E. Kasutani and A. Yamada, "The MPEG-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval," in *Proc. of ICIP*, 2001, pp. 674–677.

[8] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *TPAMI*, vol. 24, no. 4, pp. 509–522, 2002.

[9] S. Won, D. Park, and S. Park, "Efficient use of mpeg-7 edge histogram descriptor," *ETRI Journal*, vol. 24, no. 1, pp. 23–30, 2002.

[10] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *TPAMI*, vol. 19, no. 5, pp. 530–535, 1997.

[11] R. Ji, L.-Y. Duan, J. Chen, H. Yao, T. Huang, and W. Gao, "Learning compact visual descriptor for low bit rate mobile landmark search," in *Proc. of IJCAI*, 2011, pp. 2456–2463.

[12] R. Ji, L.-Y. Duan, J. Chen, H. Yao, J. Yuan, Y. Rui, and W. Gao, "Location discriminative vocabulary coding for mobile landmark search," *IJCV*, vol. 96, no. 3, pp. 290–314, 2012.

[13] J. Lin, L.-Y. Duan, J. Chen, R. Ji, S. Luo, and W. Gao, "Learning multiple codebooks for low bit rate mobile visual search," in *Proc. of ICASSP*, 2012, pp. 933–936.

[14] Y.-C. Su, T.-H. Chiu, Y.-Y. Chen, C.-Y. Yeh, and W. H. Hsu, "Enabling low bitrate mobile visual recognition: a performance versus bandwidth evaluation," in *Proc. of MM*, 2013, pp. 73–82.

[15] H. Qi, M. Stojmenovic, K. Li, Z. Li, and W. Qu, "A low transmission overhead framework of mobile visual search based on vocabulary decomposition," *TMM*, vol. 16, no. 7, pp. 1963–1972, 2014.

[16] A. Hartl, D. Schmalstieg, and G. Reitmayr, "Client-side mobile visual search," in *Proc. of VISAPP*, 2014.

[17] D. Gálvez-López and J. D. Tardós, "Real-time loop detection with bags of binary words," in *Proc. of IROS*, 2011, pp. 51–58.

[18] ——, "Bags of binary words for fast place recognition in image sequences," *TRO*, vol. 28, no. 5, pp. 1188–1197, 2012.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc. of ICCV*, 2011, pp. 2564–2571.

[20] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proc. of ICCV*, 2011, pp. 2548–2555.

[21] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Proc. of CVPR*, 2012, pp. 510–517.

[22] P. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *Proc. of BMVC*, 2013.

[23] G. Levi and T. Hassner, "Latch: Learned arrangements of three patch codes," in *Proc. of WACV*, 2016.

[24] A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, J. Ascenso, and R. Cilla, "Evaluation of low-complexity visual feature detectors and descriptors," in *Proc. of DSP*, 2013, pp. 1–7.

[25] Y. Uchida, "Local feature detectors, descriptors, and image representations: A survey," *arXiv:1607.08368*, 2016.

[26] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *TPAMI*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[27] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *IJCV*, vol. 60, no. 1-2, pp. 43–72, Nov. 2005.

[28] S. Gauglitz, S. Barbara, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *IJCV*, vol. 94, no. 3, pp. 335–360, 2011.

[29] O. Miksik and K. Mikolajczyk, "Evaluation of local detectors and descriptors for fast feature matching," in *Proc. of ICPR*, 2012.

[30] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Proc. of ECCV*, 2012, pp. 759–773.

[31] S. Bianco, R. Schettini, D. Mazzini, and D. P. Pau, "Quantitative review of local descriptors for visual search," in *Proc. of ICCE*, 2013.

[32] J. Chao, A. Al-Nuaimi, G. Schroth, and E. Steinbachs, "Performance comparison of various feature detector-descriptor combinations for content-based image retrieval with jpeg-encoded query images," in *Proc. of MMSP*, 2013, pp. 29–34.

[33] L. A. S. Filipe, "A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset," in *Proc. of VISAPP*, 2014, pp. 476–483.

[34] S. Bianco, D. Mazzini, D. Pau, and R. Schettini, "Local detectors and compact descriptors for visual search: A quantitative comparison," *Digital Signal Processing*, vol. 44, pp. 1–13, 2015.

[35] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of AVC*, 1988, pp. 147–151.

[36] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proc. of ICCV*, 2001, pp. 525–531.

[37] ——, "An affine invariant interest point detector," in *Proc. of ECCV*, 2002, pp. 128–142.

[38] ——, "Scale & affine invariant interest point detectors," *IJCV*, vol. 60, no. 1, pp. 63–86, 2004.

[39] T. Lindeberg, "Indexing based on scale invariant interest points," in *Proc. of ICCV*, 1995, pp. 134–141.

[40] T. Lindeberg and J. Garding, "Shape-adapted smoothing in estimation of 3d shape curs from affine deformations of local 2-d brightness structure," *IVC*, vol. 15, no. 6, pp. 415–434, 1997.

[41] P. Beaudet, "Rotationally invariant image operators," in *Proc. of IJCPR*, 1978, pp. 579–583.

[42] T. Lindeberg, "Feature detection with automatic scale selection," *IJCV*, vol. 30, no. 2, pp. 79–116, 1998.

[43] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of CVPR*, 1999, pp. 1150–1157.

[44] ——, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[45] M. Brown and D. G. Lowe, "Invariant features from interest point groups," in *Proc. of BMVC*, 2002, pp. 1150–1157.

[46] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proc. of ECCV*, 2006, pp. 404–417.

[47] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *CVIU*, vol. 110, no. 3, pp. 346–359, 2008.

[48] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of CVPR*, 2001, pp. 511–518.

[49] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Proc. of ICCV*, 2005, pp. 1508–1515.

[50] ——, "Machine learning for high-speed corner detection," in *Proc. of ECCV*, 2006, pp. 430–443.

[51] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *TPAMI*, vol. 32, no. 1, pp. 105–119, 2010.

[52] S. Smith and J. Brady, "Susan–a new approach to low level image processing," *IJCV*, vol. 23, no. 1, pp. 45–78, 1997.

[53] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *Proc. of ECCV*, 2010.

[54] J. Koenderink and A. van Doorn, "Representation of local geometry in the visual system," *Biological Cybernetics*, vol. 55, pp. 367–375, 1987.

[55] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Proc. of CVPR*, 2012, pp. 2911–2918.

[56] T. Kobayashi, "Dirichlet-based histogram feature transform for image classification," in *Proc. of CVPR*, 2014, pp. 3278–3285.

[57] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proc. of ECCV*, 2010, pp. 778–792.

[58] P. L. Rosin, "Measuring corner properties," *CVIU*, vol. 73, no. 2, pp. 291–307, 1999.

[59] S. Winder and M. Brown, "Learning local image descriptors," in *Proc. of CVPR*, 2007.

[60] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *TPAMI*, vol. 32, no. 5, pp. 815–830, 2010.

[61] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. of ICCV*, 2003, pp. 1470–1477.

[62] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. of CVPR*, 2006, pp. 2161–2168.

[63] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. of CVPR*, 2007, pp. 1–8.

[64] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. of ECCV*, 2008, pp. 304–317.

[65] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. of ECCV SLCV Workshop*, 2004.

[66] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. of CVPR*, 2006, pp. 2169–2178.

[67] Y. Jiang, C. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *Proc. of CIVR*, 2007, pp. 494–501.

[68] M. Douze, H. Jégou, and C. Schmid, "An image-based approach to video copy detection with spatio-temporal post-filtering," *IEEE Trans. on Multimedia*, vol. 12, no. 4, pp. 257–266, 2010.

[69] Y. Uchida, M. Agrawal, and S. Sakazawa, "Accurate content-based video copy detection with efficient feature indexing," in *Proc. of ICMR*, 2011.

[70] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *Proc. of ICCV*, 2007.

[71] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[72] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proc. of CVPR*, 2005, pp. 775–781.

[73] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Proc. of CVPR*, 2008, pp. 1–8.

[74] J. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proc. of CVPR*, 1997, pp. 1000–1006.

[75] A. Mikulík, M. Perdoch, O. Chum, and J. Matas, "Learning a fine vocabulary," in *Proc. of ECCV*, 2010, pp. 1–14.

[76] ——, "Learning vocabularies over a fine quantization," *IJCV*, vol. 103, no. 1, pp. 163–175, 2013.

[77] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *IJCV*, vol. 87, no. 3, pp. 316–336, 2010.

[78] R. Tavenard, H. Jégou, and L. Amsaleg, "Balancing clusters to reduce response time variability in large scale image search," in *Proc. of CBMI*, 2011, pp. 19–24.

[79] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *Proc. of CVPR*, 2008, pp. 1–8.

[80] W. Zhao, X. Wu, and C. Ngo, "On the annotation of web videos by efficient near-duplicate search," *TMM*, vol. 12, no. 5, pp. 448–461, 2010.

[81] F. Wang, W.-L. Zhao, C.-W. Ngo, and B. Merialdo, "A hamming embedding kernel with informative bag-of-visual words for video semantic indexing," *TOMM*, vol. 10, no. 3, 2014.

[82] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *TPAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[83] Y. Uchida, K. Takagi, and S. Sakazawa, "Ratio voting: A new voting strategy for large-scale image retrieval," in *Proc. of ICME*, 2012.

[84] D. Qin, C. Wengert, and L. V. Gool, "Query adaptive similarity for large scale object retrieval," in *Proc. of CVPR*, 2013, pp. 1610–1617.

[85] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. of NIPS*, 2008, pp. 1753–1760.

[86] J. Brandt, "Transform coding for fast approximate nearest neighbor search in high dimensions," in *Proc. of CVPR*, 2010, pp. 1815–1822.

[87] L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Lp-norm idf for large scale image search," in *Proc. of CVPR*, 2013.

[88] L. Zheng, S. Wang, and Q. Tian, "Lp-norm idf for scalable image retrieval," *TIP*, vol. 23, no. 8, pp. 3604–3617, 2014.

[89] M. Murata, H. Nagano, R. Mukai, K. Kashino, and S. Satoh, "Bm25 with exponential idf for instance search," *TMM*, vol. 16, no. 6, pp. 1690–1699, 2014.

[90] Y. Uchida and S. Sakazawa, "Large-scale image retrieval as a classification problem," *CVA*, vol. 5, pp. 153–162, 2013.

[91] H. Jégou, M. Douze, and C. Schmid, "On the burstiness of visual elements," in *Proc. of CVPR*, 2009, pp. 1169–1176.

[92] M. Jain, R. Benmokhtar, P. Grosand, and H. Jégou, "Hamming embedding similarity-based image classification," in *Proc. of ICMR*, 2012.

[93] O. Chum, J. Matas, and Š. Obdržálek, "Enhancing ransac by generalized model optimization," in *Proc. of ACCV*, 2004.

[94] O. Chum and J. Matas, "Matching with prosac - progressive sample consensus," in *Proc. of CVPR*, 2005.

[95] M. Perd'och, O. Chum, and J. Matas, "Efficient representation of local geometry for large scale object retrieval," in *Proc. of CVPR*, 2009, pp. 9–16.

[96] Y. Zhang, Z. Jia, and T. Chen, "Image retrieval with geometry-preserving visual phrases," in *Proc. of CVPR*, 2011, pp. 809–816.

[97] X. Shen, Z. Lin, J. Brandt, S. Avidan, and Y. Wu, "Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking," in *Proc. of CVPR*, 2012, pp. 3013–3020.

[98] J. Wang, J. Tang, and Y. Jiang, "Strong geometrical consistency in large scale partial-duplicate image search," in *Proc. of MM*, 2013, pp. 633–636.

[99] Z. Zhong, J. Zhu, and S. C. H. Hoi, "Fast object retrieval using direct spatial matching," *TMM*, vol. 17, no. 8, pp. 1391–1397, 2015.

[100] C. Ngo, W. Zhao, and Y. Jiang, "Fast tracking of near-duplicate keyframes in broadcast domain with transitivity propagation," in *Proc. of MM*, 2006, pp. 845–854.

[101] W. L. Zhao and C. W. Ngo, "Scale-rotation invariant pattern entropy for keypoint-based near-duplicate detection," *TIP*, vol. 18, no. 2, pp. 412–423, 2009.

[102] S. S. Tsai, D. M. Chen, G. Takacs, V. Chandrasekhar, R. Vedantham, R. Grzeszczuk, and B. Girod, "Fast geometric reranking for image based retrieval," in *Proc. of ICIP*, 2010, pp. 1029–1032.

[103] Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang, "Spatial-bag-of-features," in *Proc. of CVPR*, 2010.

[104] Z. Lin and J. Brandt, "A local bag-of-features model for large-scale object retrieval," in *Proc. of ECCV*, 2010, pp. 294–308.

[105] Z. Wu, Q. Ke, M. Isard, and J. Sun, "Bundling features for large scale partial-duplicateweb image search," in *Proc. of CVPR*, 2009.

[106] Z. Wu, Q. Ke, J. Sun, and H. Y. Shum, "A multi-sample, multi-tree approach to bag-of-words image representation for image retrieval," in *Proc. of ICCV*, 2009, pp. 1992–1999.

[107] S. Zhang, Q. Huang, G. Hua, S. Jiang, W. Gao, and Q. Tian, "Building contextual visual vocabulary for large-scale image applications," in *Proc. of MM*, 2010, pp. 501–510.

[108] O. Chum, A. Mikulík, M. Perdoch, and J. Matas, "Total recall II: Query expansion revisited," in *Proc. of CVPR*, 2011, pp. 889–896.

[109] D. Qin, S. Grammeter, L. Bossard, T. Quack, and L. V. Gool, "Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors," in *Proc. of CVPR*, 2011, pp. 777–784.

[110] G. Tolias, Y. Avrithis, and H. Jégou, "To aggregate or not to aggregate: selective match kernels for image search," in *Proc. of ICCV*, 2013.

[111] S. Zhang, M. Yang, X. Wang, Y. Lin, and Q. Tian, "Semantic-aware co-indexing for image retrieval," in *Proc. of ICCV*, 2013.

[112] L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Packing and padding: Coupled multi-index for accurate image retrieval," in *Proc. of CVPR*, 2014, pp. 1947–1954.

[113] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Proc. of NIPS*, 1998, pp. 487–493.

[114] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. of CVPR*, 2007.

[115] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[116] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. of ECCV*, 2010, pp. 143–156.

[117] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *IJCV*, vol. 105, no. 3, pp. 222–245, 2013.

[118] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Proc. of CVPR*, 2010, pp. 3384–3391.

[119] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. of CVPR*, 2010, pp. 3304–3311.

[120] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *TPAMI*, vol. 34, no. 9, pp. 1704–1716, 2012.

[121] B. Klein, G. Lev, G. Sadeh, and L. Wolf, "Associating neural word embeddings with deep image representations using fisher vectors," in *Proc. of CVPR*, 2015, pp. 4437–4446.

[122] J. Sánchez and J. Redolfi, "Exponential family fisher vector for image classification," *PRL*, vol. 59, pp. 26–32, 2015.

[123] G. Amato, F. Falchi, and L. Vadicamo, "Aggregating binary local descriptors for image retrieval," *arXiv:1608.00813*, 2016.

[124] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep fisher networks for large-scale image classification," in *Proc. of NIPS*, 2013.

[125] F. Perronnin and D. Larlus, "Fisher vectors meet neural networks: A hybrid classification architecture," in *Proc. of CVPR*, 2015.

[126] V. Chandrasekhar, J. Lin, O. Morère, H. Goh, and A. Veillard, "A practical guide to CNNs and fisher vectors for image instance retrieval," *Signal Processing*, vol. 128, pp. 426–439, 2016.

[127] O. Morère, A. Veillard, J. Lin, J. Petta, V. Chandrasekhar, and T. Poggio, "Group invariant deep representations for image instance retrieval," *arXiv:1601.02093*, 2016.

[128] H. Jégou and O. Chum, "Negative evidences and co-occurrences in image retrieval: the benefit of pca and whitening," in *Proc. of ECCV*, 2012.

[129] R. Arandjelović and A. Zisserman, "All about VLAD," in *Proc. of CVPR*, 2013.

[130] J. Delhumeau, P. Gosselin, H. Jégou, and P. Pérez, "Revisiting the VLAD image representation," in *Proc. of MM*, 2013, pp. 653–656.

[131] H. Jégou and A. Zisserman, "Triangulation embedding and democratic aggregation for image search," in *Proc. of CVPR*, 2014.

[132] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod, "Residual enhanced visual vector as a compact signature for mobile visual search," *Signal Processing*, vol. 93, no. 8, pp. 2316–2327, 2013.

[133] E. Spyromitros-Xioufis, S. Papadopoulos, I. Kompatsiaris, G. Tsoumakas, and I. Vlahavas, "A comprehensive study over VLAD and product quantization in large-scale image retrieval," *TMM*, vol. 16, no. 6, pp. 1713–1728, 2014.

[134] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of NIPS*, 2012.

[135] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Proc. of ECCV*, 2014, pp. 584–599.

[136] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. of CVPRW*, 2014, pp. 512–519.

[137] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Visual instance retrieval with deep convolutional networks," *arXiv:1412.6574*, 2014.

[138] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, "Deep learning for content-based image retrieval: A comprehensive study," in *Proc. of MM*, 2014, pp. 157–166.

[139] J. L. Long, N. Zhang, and T. Darrell, "Do convnets learn correspondence?" in *Proc. of NIPS*, 2014, pp. 1601–1609.

[140] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. of ECCV*, 2014.

[141] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proc. of CVPR*, 2015.

[142] Y. Liu, Y. Guo, and S. W. andMichael S. Lew, "Deepindex for accurate and efficient image retrieval," in *Proc. of ICMR*, 2015, pp. 43–50.

[143] L. Xie, R. Hong, B. Zhang, and Q. Tian, "Image classification and retrieval are ONE," in *Proc. of ICMR*, 2015, pp. 3–10.

[144] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. of CVPR*, 2008, pp. 1–8.

[145] J. Y.-H. Ng, F. Yang, and L. S. Davis, "Exploiting local features from deep networks for image retrieval," in *Proc. of CVPRW*, 2015, pp. 53–61.

[146] A. Babenko and V. Lempitsky, "Aggregating deep convolutional features for image retrieval," in *Proc. of ICCV*, 2015, pp. 1269–1277.

[147] M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronnin, and C. Schmid, "Local convolutional features with unsupervised training for image retrieval," in *Proc. of ICCV*, 2015, pp. 91–99.

[148] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Proc. of NIPS*, 2014.

[149] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. of ICCV*, 2015.

[150] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. of CVPR*, 2015.

[151] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *Proc. of CVPR*, 2015.

[152] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "TILDE: A temporally invariant learned detector," in *Proc. of CVPR*, 2015.

[153] K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit, "Learning to assign orientations to feature points," in *Proc. of CVPR*, 2016.

[154] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," *arXiv:1603.09114*, 2016.

[155] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv:1412.6115*, 2014.

[156] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *Proc. of ICLR*, 2016.

[157] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *arXiv:1603.05279*, 2016.

[158] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. of ICLR*, 2016.

[159] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *Proc. of ISCA*, 2016.

[160] P. Alcantarilla, A. Bartoli, and A. Davison, "Kaze features," in *Proc. of ECCV*, 2012.

[161] X. Yang and K. Cheng, "Ldb: An ultra-fast feature for scalable augmented reality on mobile devices," in *Proc. of ISMAR*, 2012, pp. 49–57.

[162] X. Yang and K.-T. T. Cheng, "Local difference binary for ultrafast and distinctive feature description," *TPAMI*, vol. 36, no. 1, pp. 188–194, 2014.

[163] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. of NIPS*, 2009.

[164] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. of CVPR*, 2010, pp. 3424–3431.

[165] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. of CVPR*, 2011, pp. 817–824.

[166] M. Ambai and Y. Yoshida, "Card: Compact and real-time descriptors," in *Proc. of ICCV*, 2011.

[167] Y. Lee, J. Heo, and S. Yoon, "Quadra-embedding: Binary code embedding with low quantization error," in *Proc. of ACCV*, 2012.

[168] G. Irie, Z. Li, X. Wu, and S. Chang, "Locally linear hashing for extracting non-linear manifolds," in *Proc. of CVPR*, 2014.

[169] V. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. of CVPR*, 2015.

[170] Y. Uchida and S. Sakazawa, "Image retrieval with fisher vectors of binary features," in *Proc. of ACPR*, 2013.

[171] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. of ISMIR*, 2002, pp. 107–115.

[172] X. Anguera, A. Garzon, and T. Adamek, "Mask: Robust local features for audio fingerprinting," in *Proc. of ICME*, 2012.

[173] A. Juan and E. Vidal, "Bernoulli mixture models for binary images," in *Proc. of ICPR*, 2004, pp. 367–370.

[174] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. of VLDB*, 1999, pp. 518–529.

[175] M. Muja and D. G. Lowe, "Fast matching of binary features," in *Proc. of CRV*, 2012.

[176] W. Zhou, Y. Lu, H. Li, and Q. Tian, "Scalar quantization for large scale image search," in *Proc. of MM*, 2012.

[177] S. McCann and D. G. Lowe, "Local naive bayes nearest neighbor for image classification," in *Proc. of CVPR*, 2012.

[178] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. of NIPS*, 2004, pp. 1601–1608.

[179] T. Roelleke and J. Wang, "Tf-idf uncovered: A study of theories and probabilities," in *Proc. of SIGIR*, 2008, pp. 435–442.

[180] P. Turcot and D. G. Lowe, "Better matching with fewer features: The selection of useful features," in *Proc. of WS-LAVD*, 2009.

[181] Z. Wang, Q. Zhao, D. Chu, F. Zhao, and L. J. Guibas, "Select informative features for recognition," in *Proc. of ICIP*, 2011.

[182] K. Matsuzaki, Y. Uchida, S. Sakazawa, and S. Satoh, "Local feature reliability measure using multiview synthetic images for mobile visual search," in *Proc. of ACPR*, 2015.

[183] L. Bo and C. Sminchisescu, "Efficient match kernels between sets of features for visual recognition," in *Proc. of NIPS*, 2009.

[184] G. Tolias, Y. Avrithis, and H. Jégou, "Image search with selective match kernels: Aggregation across single and multiple images," *IJCV*, vol. 116, no. 3, pp. 247–261, 2016.

[185] Y. Uchida and S. Sakazawa, "Accurate feature matching and scoring for re-ranking image retrieval," in *Proc. of ICME*, 2013.

[186] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *JACM*, vol. 45, no. 6, pp. 891–923, 1998.

[187] A. Andoni, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. of FOCS*, 2006, pp. 459–468.

[188] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. of VISAPP*, 2009, pp. 331–340.

[189] ——, "Scalable nearest neighbor algorithms for high dimensional data," *TPAMI*, vol. 36, no. 11, pp. 2227–2240, 2014.

[190] S. Brin, "Near neighbor search in large metric spaces," in *Proc. of VLDB*, 1995.

[191] T. Trzcinski, V. Lepetit, and P. Fua, "Thick boundaries in binary space and their influence on nearest-neighbor search," *PRL*, vol. 33, no. 16, pp. 2173–2180, 2012.

Publications

Publications related to the thesis

Journal papers

[1] Y. Uchida, S. Sakazawa, and S. Satoh, “Image Retrieval with Fisher Vectors of Binary Features,” in *ITE Trans. on MTA*, 2016 (accepted).

International conference

[2] Y. Uchida, S. Sakazawa, and S. Satoh, “Binary Feature-based Image Retrieval with Effective Indexing and Scoring,” in *Proc. of GCCE*, 2014. (oral)

[3] Y. Uchida and S. Sakazawa, “Image Retrieval with Fisher Vectors of Binary Features,” in *Proc. of ACPR*, 2013. (oral)

Publications non-related to the thesis

Journal papers

[4] S. Kasamwattanarote, Y. Uchida, and S. Satoh, “Query Bootstrapping: A Visual Mining based Query Expansion," in *IEICE Trans. on Information and Systems*, Vol. E99D, No. 2, 2016.

[5] 小林亜令, 松本正明, 内田祐介, 土井渉, 松崎康平, 加藤晴久, “端末とサーバによるハイブリッド大規模画像検索システムとその応用," 映情学誌, Vol. 69, No. 1, pp. 11–16, 2015.

[6] Y. Uchida and S. Sakazawa, “Large-Scale Image Retrieval as a Classification Problem," in *IPSJ Trans. on CVA*, Vol. 8, No. 4, pp. 1130-1139, 2013.

[7] Y. Uchida, K. Takagi, and S. Sakazawa, “Optimized Codebook Construction and Assignment for Product Quantization-based Approximate Nearest Neighbor Search," in *IPSJ Trans. on CVA*, vol. 4, pp. 108–118, 2012.

[8] 内田祐介, 橋本真幸, 川田亮一, “BoFを利用した映像検索における索引規模削減手法," 信学論, Vol. J94D, No. 1, pp. 416–420, 2011.

[9] Y. Uchida, K. Takagi, and R. Kawada, “Quantization-Based Approximate Nearest Neighbor Search with Optimized Multiple Residual Codebooks," in *IEICE Trans. on Information and Systems*, Vol. E94D, No. 7, pp. 1510–1514, 2011.

[10] 内田祐介, 橋本真幸, 米山暁夫, 川田亮一, “輝度重心に基づくバイナリ特徴量の適応的照合による高速・高精度な同一映像検出," 信学論, Vol. J93D. No. 9, pp. 1714–1716, 2010.

International conference (refereed)

[11] K. Matsuzaki, Y. Uchida, S. Sakazawa, and S. Satoh, “Geometric Verification Using Semi-2D Constraints for 3D Object Retrieval," in *Proc. of ICPR*, 2016. (accepted)

[12] K. Matsuzaki, Y. Uchida, S. Sakazawa, and S. Satoh, “Local Feature Reliability Measure Using Multiview Synthetic Images for Mobile Visual Search," in *Proc. of ACPR*, 2015.

[13] Y. Uchida and S. Sakazawa, “Accurate Feature Matching and Scoring for Reranking Image Retrieval Results," in *Proc. of ICME*, 2013.

[14] Y. Nagai, Y. Uchida, E. Myodo, and S. Sakazawa, “A color transformation method based on color theme that takes constraints on color ratio and spatial coherence into consideration,” in *Proc. of ICIP*, 2013.

[15] Y. Uchida, K. Takagi, and S. Sakazawa, “An Alternative to IDF: Effective Scoring for Accurate Image Retrieval with Non-Parametric Density Ratio Estimation," in *Proc. of ICPR*, 2012.

[16] Y. Uchida, K. Takagi, and S. Sakazawa, “Ratio Voting: A New Voting Strategy for Large-Scale Image Retrieval," in *Proc. of ICME*, 2012. (oral)

[17] Y. Uchida, K. Takagi, and S. Sakazawa, “Fast and Accurate Content-Based video Copy Detection Using Bag-of-Global Visual Features," in *Proc. of ICASSP*, 2012.

[18] Y. Uchida, M. Agrawal, and S. Sakazawa, “Accurate Content-Based Video Copy Detection with Efficient Feature Indexing," in *Proc. of ICMR*, 2011. (oral)

[19] Y. Uchida, M. Hashimoto, and R. Kawada, “Fast and Robust Content-Based Copy Detection Based on Quadrant of Luminance Centroid and Adaptive Feature Comparison," in *Proc. of ICIP*, 2010. (oral)

International conference (non-refereed)

[20] Y. Uchida, K. Takagi, and S. Sakazawa, “KDDI Labs at TRECVID 2011: Content-Based Copy Detection," in *Proc. of TRECVID*, 2011.

[21] Y. Uchida, S. Sakazawa, M. Agrawal, and M. Akbacak, “KDDI Labs and SRI International at TRECVID 2010: Content-Based Copy Detection," in *Proc. of TRECVID*, 2010.

Domestic conference (refereed)

[22] 内田祐介, 酒澤茂之, "複数コードブックを用いた直積量子化による近似最近傍探索手法と特定物体認識への応用," 画像の認識・理解シンポジウム, 2011. (single oral)

Domestic conference (non-refereed)

[23] 内田祐介, 酒澤茂之, "Image Retrieval with Fisher Vectors of Binary Features," 画像の認識・理解シンポジウム, 2014.

[24] 内田祐介, 酒澤茂之, "クラス分類問題としての画像検索," 画像の認識・理解シンポジウム, 2013.

[25] 内田祐介, 酒澤茂之, "Local NBNNの画像検索への適用に関する一検討," 映像情報メディア学会年次大会, 2013.

[26] 内田祐介, 酒澤茂之, "バイナリ特徴のためのフィッシャーベクトルに関する一検討," 電子情報通信学会総合大会, 2013.

[27] 内田祐介, 酒澤茂之, "特定物体認識における画像間の幾何検証の高精度化に関する一検討," 映像情報メディア学会冬季大会, 2012.

[28] 内田祐介, 酒澤茂之, "特定物体認識のための最近傍密度比推定に基づくスコアリング手法," 画像の認識・理解シンポジウム, 2012.

[29] 内田祐介, 酒澤茂之, "特定物体認識のための最近傍密度推定に関する一検討," 映像情報メディア学会年次大会, 2012.

[30] 内田祐介, 酒澤茂之, "大域・局所画像特徴および音響特徴を用いた高精度なコピー検出手法," パターン認識・メディア理解研究会, 信学技報, 2011.

[31] 内田祐介, 酒澤茂之, "局所特徴の時間的バースト性を考慮した準同一映像検出," 電子情報通信学会総合大会, 2011.

[32] 内田祐介, 高木幸一, 酒澤茂之, "近似最近傍探索のための直積量子化コードブック作成手法に関する一検討," 映像情報メディア学会年次大会, 2011.

[33] 内田祐介, 高木幸一, 酒澤茂之, "BoVWを用いた特定物体認識における投票関数に関する一考察," 情報科学技術フォーラム, 2011.

[34] 内田祐介, 橋本真幸, 川田亮一, "ショット境界のbi-gram表現による同一映像検索手法に関する一検討," 電子情報通信学会総合大会, 2010.

[35] 内田祐介, 菅野勝, 橋本真幸, 米山暁夫, "カラーレイアウト記述子を利用したコピー検出手法の性能評価," パターン認識・メディア理解研究会, 信学技報, 2009.

[36] 内田祐介, 橋本真幸, 米山暁夫, 川田亮一, "画像マッチングとカメラ追跡によるカメラポインタの性能改善," 映像情報メディア学会冬季大会, 2009.

[37] 内田祐介, 橋本真幸, 米山暁夫, "Iフレームを用いたショット境界検出の高速化に関する性能評価," 情報科学技術フォーラム, 2009.

[38] 内田祐介, 橋本真幸, 米山暁夫, "バイナリ特徴量を用いた高速同一映像断片探索に関する一検討," 映像メディア処理シンポジウム, 2009.

[39] 内田祐介, 橋本真幸, 米山暁夫, "Iフレームを用いたショット境界検出の高速化に関する一検討," 映像情報メディア学会年次大会, 2009.

[40] 内田祐介, 加藤晴久, 上野智史, 橋本真幸, 米山暁夫, "アフィン不変領域抽出のためのSIFT拡張の検討," 電子情報通信学会総合大会, 2009.

[41] 内田祐介, 菅野勝, 米山暁夫, "カラーレイアウトを利用した違法コピー検出における投票に関する一考察," 映像情報メディア学会冬季大会, 2008.

[42] 内田祐介, 菅野勝, 米山暁夫, "カラーレイアウトに基づく違法コピー検出手法," 映像メディア処理シンポジウム, 2008.

[43] 内田祐介, 加藤晴久, 米山暁夫, "前景・背景分離を利用したタイムラプスビデオに関する一検討," 映像情報メディア学会年次大会, 2008.

[44] 内田祐介, 加藤晴久, 宮地悟史, 米山暁夫, "動画のエラー伝播特性を考慮した不均一誤り保護の一検討," 電子情報通信学会総合大会, 2008.

Technical descriptions

[45] Y. Uchida and S. Sakazawa, "Near-Duplicate Image Retrieval as a Classification Problem," in *IEEE COMSOC MMTC E-Letter*, vol. 7, no. 7, 2012.

[46] 内田祐介, 酒澤茂之, "大規模特定物体認識の最新動向," 信学会誌, Vol. 93, No. 3, pp. 207–213, 2013.

[47] 内田祐介, 酒澤茂之, "大規模特定物体認識技術およびその最新研究事例," Vol. 24, No. 12, pp. 61–68, 2013.

Awards

[48] 映像情報メディア学会技術振興賞, 2013.

[49] ACPR 2013 Best Paper Award, 2013.

[50] FITヤングリサーチャー賞, 2011.

[51] 電子情報通信学会学術奨励賞, 2010.