석사 학위논문

Master's Thesis

초 고차원 표현법 기반의 이미지 검색을 위한 코드워드 중요성 학습 방법

Learning Codeword Characteristics for Image Retrieval Using Very High Dimensional Bag-of-Words Representation

유 동 근 (兪 同 根 Yoo, Donggeun)

전기 및 전자공학과

Department of Electrical Engineering

KAIST

2013

초 고차원 표현법 기반의 이미지 검색을 위한 코드워드 중요성 학습 방법

Learning Codeword Characteristics for Image Retrieval
Using Very High Dimensional Bag-of-Words Representation

Learning Codeword Characteristics for Image Retrieval Using Very High Dimensional Bag-of-Words Representation

Advisor : Professor Kweon, In So

by

Yoo, Donggeun

Department of Electrical Engineering

KAIST

A thesis submitted to the faculty of KAIST in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Department of Electrical Engineering . The study was conducted in accordance with Code of Research Ethics[1].

2013. 1. 2.

Approved by

Professor Kweon, In So

[Advisor]

[1]Declaration of Ethical Conduct in Research: I, as a graduate student of KAIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

초 고차원 표현법 기반의 이미지 검색을 위한 코드워드 중요성 학습 방법

유 동 근

위 논문은 한국과학기술원 석사학위논문으로
학위논문심사위원회에서 심사 통과하였음.

2012년 12월 20일

MEE
20113372

유 동 근. Yoo, Donggeun. Learning Codeword Characteristics for Image Retrieval Using Very High Dimensional Bag-of-Words Representation. 초 고차원 표현법 기반의 이미지 검색을 위한 코드워드 중요성 학습 방법. Department of Electrical Engineering . 2013. 54p. Advisor Prof. Kweon, In So. Text in English.

ABSTRACT

In this thesis, we address three problems of the *tf-idf* weighting, the hierarchical scoring and intra-class key words in the image retrieval using very large dimensional Bag-of-Words (BoW) representation. The *tf-idf* weighting method, which is a commonly used codebook weighting scheme, is usually understood to improve retrieval performance however its degree is not too significant. Rather, it sometimes brings a problem of worsening precision. The hierarchical scoring, which is commonly used in hierarchical codebook, the precision improvement differs depending on the dataset as the number of levels that are considered in scoring gets larger. Intra-class key words, which represent their class most well, have not taken into consideration in BoW representation based image retrieval because of its too high dimensionality. Despite different classes have different key-word, only a same weight or standard is applied to every image classes. To overcome these three different problems, we suggest a new codewords weighting method preserving the independence model of BoWs representation that codewords occur independently in one image. In the problems of *tf-idf* weighting and the hierarchical scoring, the proposed method only focuses on improving the algorithms without using any extra cue besides two types of signature, document frequency used for *tf-idf* weighting and the level for hierarchical scoring. Since the document frequency and the level are related with inter-class discriminability, we define the two values as key-signatures. In the problem of considering intra-class key words, the proposed method gives relevant weight to codewords according to its statistical appearance within a class. Since an intra-class variance of frequencies of a codeword is related with its intra-class importance, we define that kind of value as another key-signature. We also define a function, called Weight Mapping Function (WMF), that maps a weight value from a key-signature. In order to obtain optimal WMF shapes for each key-signature type, we learn the WMFs using randomly sampled training data by means of optimization method. From the WMF of document frequency or level, we produce a global weight vector for every classes, in the other hand, we produce intra-class weight vectors for each class from the WMF of intra-class variance. We conducted experiments with UKbench and OXford5K dataset, and our approach outperformed the tf-idf weighting and hierarchical scoring, without extra cue and additional computation in on-line stage. The proposed intra-class weighting method also shows noticeable improvement on UKbench dataset.

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Information retrieval has grown mainly with text retrieval, which finds desired information from the text queries. We are now provided with tremendous amount of image data as multimedia technology has evolved and the number of the users has increased drastically. Along with the development of technology to handle web-scale data, technologies to effectively index or search not only text data but also image or video data have become important. In the year of 2010, Facebook had stored over 260 billions images and one billion new images are uploaded for a week, approximately 60 terabytes [44]. In other words, one million images were uploaded to Facebook server per a second at peak. In addition to that, the demand to retrieve visual information by visual query is increasing because it is hard to describe visual information by text. For this reason, image retrieval is currently one of active and vibrant research topics in computer vision field.

The goal of image retrieval is to retrieve a set of images that have exactly the same object in a query image from a given image database as shown in Figure 1.1. The retrieved images are expected to contain the same object or scene with various transformations and time in which the images taken. It is, in a sense, same with image classification or object detection in terms of recognizing which object exists in a query image. However, there are obvious differences between image retrieval and the other as shown in Table 1.1. Image retrieval is different from image classification problems in terms of output, the former gives a set of images containing the same objects of a query, on the other hand the latter returns a class label of the query among pre-defined classes. Furthermore, image retrieval system does not have pre-trained classifiers while image classification has. In an actual scenario of image retrieval, object categories in the web image database are impossible to be defined since only duplicated objects can be categorized into the same class. For this reason, an image retrieval system stores only image vectors while an image classification system should store finite classifiers of pre-defined object classes. Likewise, image retrieval is different from the object detection problem even though they have same goal in terms of detecting the same objects. Object detection problem has pre-trained detectors to detect specified object categories such as faces, cars, humans, et cetera. In short, image retrieval system does not have pre-trained classifiers or detectors but an image database because its goal is to detect the same object of an arbitrary query image. We do not know which kind of objects a query image has and the database images do not have theirs category labels. What users

	Image retrieval	Image classification	Object detection
Output	Images containing same object	Label of query	Images containing same object
Database	Images vectors	Classifiers	Detectors
# class	Infinite	Finite	Finite

Table 1.1: Comparison of image retrieval with image classification and object detection.

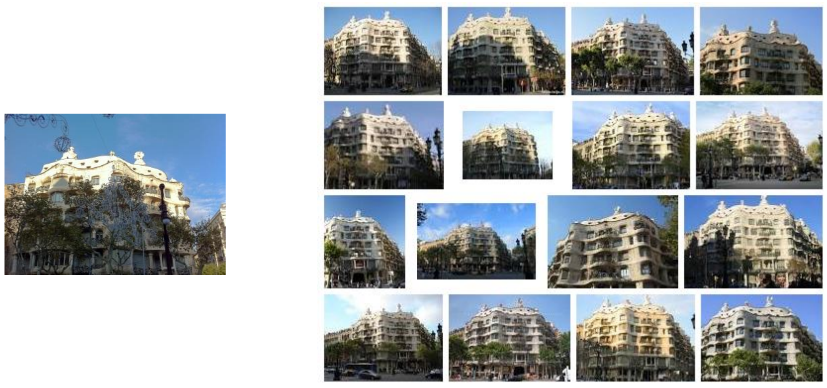


Figure 1.1: An example of image retrieval. From a query image (left), an image retrieval system returns a set of images (right) containing the exactly same object of the query.

only expect to image retrieval system is that the system gives a relevant set of images that contains the same object or scene with different view points, time, illuminations.

An image retrieval system can be used for the various types of applications. This system is very meaningful itself, because it satisfies people's desire to retrieve images or videos containing the same objects or scenes they already have in their query [2]. It can be used for commercially as an product search engine as well if database is composed of product images which have its product identities [25, 26]. Likewise, [27] retrieved visually similar clothing from a query containing clothes a human have on. Other applications of image retrieval dealing with large-scale database is enhancing some vision technologies which are not yet robust enough [28, 32, 29, 30, 31]. For example, Hays and Efros [28] improved in-paining technology, which erases user defined regions in an image and fill the regions with some visual contents, by retrieving visually similar images from large-scale database. They filled the target region with contents of the retrieved images. As another example, Torralba et al. [32] showed that we can recognize well scenes only with tiny image descriptors and image retrieval system in large-scale database (80M images). Moreover, Agarwal et al. [29, 30, 31] reconstructed a 3D model of Rome in a day with the help of image retrieval system.

Chapter 2. Retrieval System

In general, image retrieval system can be divided into two major parts. One is image description part that represents an image as a vector and the other is a part of data structure that stores large number of image vectors as a database efficiently. Of course the data structure should be very suitable to search the database very fast and accurately as well. Most image retrieval system uses Bag-of-Words representation as an image description method and inverted file as an efficient data structure [2, 6, 8, 3, 15, 14, 13, 10, 20, 9, 23, 22, 5, 33, 17].

2.1 Image Description

Most of the retrieval system is based on Bag-of-Words(BoW) model [2]. Literally, BoW means a bag containing words in a document. The description method that describes images based on BoW model is denoted as BoW representation. The idea of BoW representation in image retrieval originally started from the text retrieval, which describes a text document by the proportion of the word counts in a document. A document is vectorized to a histogram, called BoW histogram, in which a bin of a word has a value of frequency of the word (term frequency) as shown in Figure 2.1. With a pair of BoW histograms, we can compare a pair of documents by computing a distance between them. BoW model in text domain was firstly applied to image domain by Sivic et al [2]. In the BoW model of image domain, as shown in Table 2.1, an image correspond to a document of text domain. A visual dictionary correspond to a text dictionary, and words occurred in a document correspond to visual words in an image. [2] proposed the new concept of the visual words and the visual dictionary, called codebook, because intrinsically an image does not have the concept of word. In order to represent an image as a set of visual words, we consider a local descriptor in an image as a visual word because local description, which describes multiple local regions in an image not a whole image region, is robust to occlusion, articulation, background clutter and intra-class variations. Local descriptors are often computed in an image by a procedure of local region detection [1, 39, 38, 46, 47] and local region description [1, 37, 35, 36, 33, 34].

Local Region Detection In local region detection stage, we detect interesting local regions [1, 39, 38, 46, 47] or points [48, 49, 50]. One of important condition of a local region is that it should contain information distinctive

Text domain	Image domain
Text document	Image
Text dictionary	Visual dictionary (codebook)
Text word	Visual word

Table 2.1: Correspondence between text domain and image domain in the BoW model.

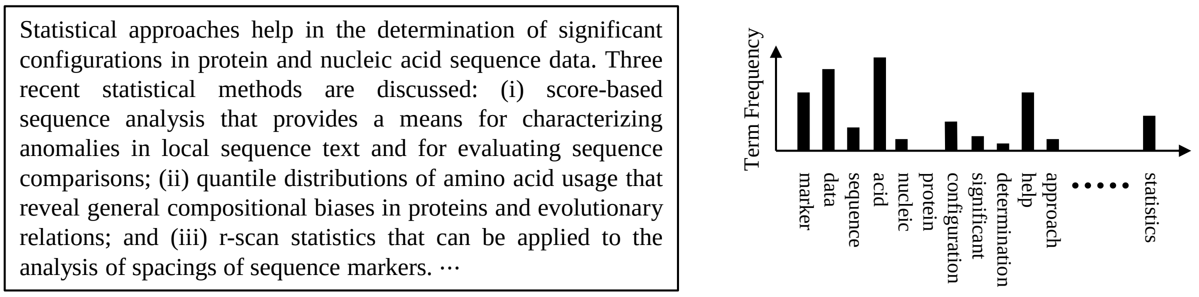

Statistical approaches help in the determination of significant configurations in protein and nucleic acid sequence data. Three recent statistical methods are discussed: (i) score-based sequence analysis that provides a means for characterizing anomalies in local sequence text and for evaluating sequence comparisons; (ii) quantile distributions of amino acid usage that reveal general compositional biases in proteins and evolutionary relations; and (iii) r-scan statistics that can be applied to the analysis of spacings of sequence markers. ⋯

Figure 2.1: An example of BoW histogram of a text document. A text document (left) is described by frequencies of terms, called BoW histogram (right).

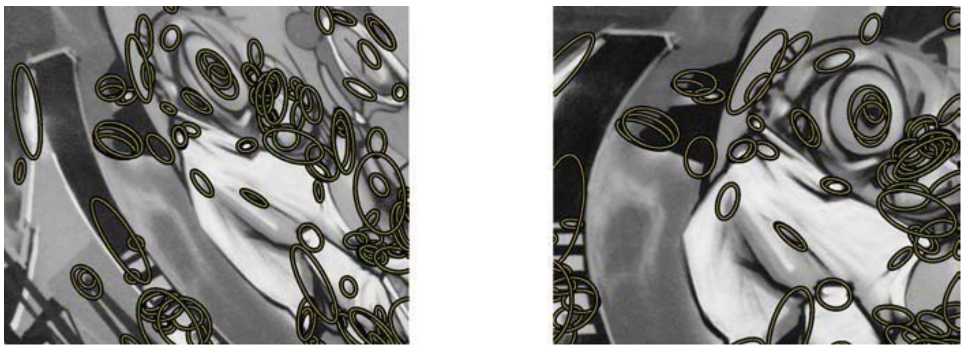


Figure 2.2: Examples of local region detection using Hessian-Affine detector [39]. Yellow ellipses denote the detected regions. The two images are in relation of affine transformation caused by different viewpoint.

from the other local regions. In order to satisfying the conditions, we use information of edges or corners in an image because flat regions do not have any information of shape. One another important condition is repeatability. If a region or a point of an object is detected in an image, a corresponding part of another image containing the same object should be detected. Detectors should guarantee repeatability as much as possible because the same objects or scenes can be taken from various situations, which cause photometric and geometric transformations. Photometric transformations mean photometric differences caused by noise, blur or illumination changes, and geometric transformations means geometric changes caused by translations, rotations or different viewpoints. Figure 2.2 shows an real example [39] of repeatability in a situation of geometric transformation. The left image is transformed to the right image because of different viewpoint. Nevertheless, in principle, the corresponding detected regions of the left image also detected in the right image. At times, we do not use detectors, instead, we define local regions or points uniformly from pre-defined step size, when we only want to consider global distribution of color [33, 34] or textures [36, 45]. Particularly, in the case of image classification problems, we do not care repeatability because it has too large intra-class variance. On the other hand, in the case of image retrieval problems, repeatability is one of significant elements to improve search performance because it should detect exactly the same object of query from database.

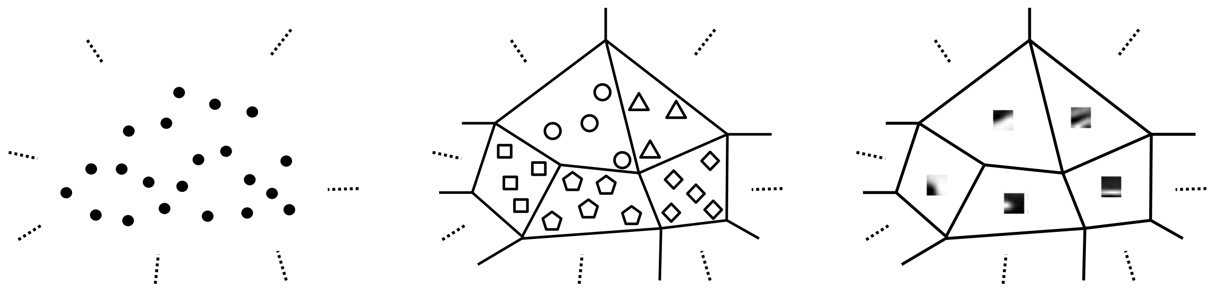


Figure 2.3: Illustrations of codebook generation. Left: Local descriptors (black dots) are scattered in their space. Center: Local descriptors are clustered by k-means clustering. Right: A centroid of a quantized cell (Voronoi cell) becomes a codeword. A set of every codewords denotes a codebook. Each codeword is visualized as each squared graphic.

Local Region Description After detecting interesting local regions or points on an image, we describe them using pixel intensities in order that the described results are invariant to several types of transformation as well. We use various types of local description methods such as color [33, 34], boundaries [35], texture [36] and gradient [1, 37]. [33, 34] describe points detected using color vectors from various color spaces, [35] using the contour information of object boundaries, and [36] using edge responses from various types of filter, called filter bank. The gradient based description method [1, 37], most commonly used in image retrieval, describes local regions detected using gradient vectors in order to make it invariant to illumination and rotation changes. In the procedure of [1, 37], we normalize the size of local region before description in order to make it invariant to scale changes.

Codebook Generation We then create a codebook which is composed of codewords by quantizing the space of local descriptor that extracted in training images as shown in Figure 2.3. Here a codebook denote a visual dictionary, likewise, a codeword denote a visual word as shown in Table 2.1. In order to quantize the descriptor space, k-means clustering method is generally used with pre-defined number of clusters (k) that means codebook size. We can consider each centroid of quantized cell as a codeword and we call a set of every codewords codebook. In image retrieval problems, the error caused by quantization process majorly harms retrieval performance because the error gives rise to miss-matches between local descriptors. For this reason, generally, we set large k when we conduct k-means clustering, from 10k to 10M, in order to reduce the error as far as possible.

BoW Representation We extract local descriptors from an image, after that, we assign a local descriptors to a codeword based on minimum distances between the descriptor and every codewords. A single image, now, can be described by how many times each codeword appears as shown in Figure 2.4, in other words, we can represent an image as a codeword frequency vector denoted by BoW histogram. As a result, like a text document, an image becomes a visual document composed of visual words, in other words, a bag which contains visual words (Bag-of-Words representation). When we compare a pair of images, we compute a distance between the pair of BoW histograms. The smaller distance two images have, the more similar images they become.

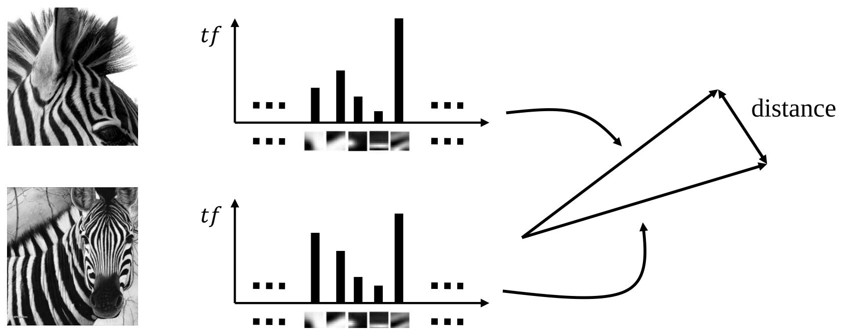


Figure 2.4: Illustrations of Bag-of-Word representation. Images (left) can be represented as BoW histograms (center) using a codebook. A pair of images is compared based on a distance (right) between the two BoW histograms.

2.2 Data Structure

Nowadays the size of database is becoming larger and larger, for that reason, it is becoming more and more difficult to handle the tremendous amount of image data. In order to compactly store the database composed of BoW histograms and database search with high speed, data structure of the image retrieval system is another major issue. When we consider a database containing 1M images and a codebook size of 1M, if we directly store the raw BoW histograms, total size of the database is 8Tbytes (1M$\times$1M$\times$8bytes) which is infeasible. In addition to that, if we naively use exhaustive database search, a query time requires one tera times of multiplying operation of floating point.

Most image retrieval systems based on high dimensional BoW representation use the inverted file structure [2, 6] which also originally comes from text retrieval system. Search engine using the Inverted file calculate very fast and efficiently exact l_p-distances between a BoW histogram of a query and that of database documents. A BoW histogram stores codeword frequencies of an image, and it denote forward file. Contrary to the forward file, in the inverted file, literally, a codeword stores indices of image files which contain the codeword as shown in Figure 2.5. With this structure, we can only considers codewords occurred in a query image, not every codewords in the case of forward file structure. For example, in Figure 2.5, if the $(n-1)$-th codeword is appeared only in a query image, we only considers the image of id 1 not the other images in the database. In short, inverted file enables both compact memory usage and fast database search without any accuracy loss because it stores only image indices and considers only images containing codewords appeared in a query image.

When it combined with image retrieval system, inverted file gives another powerful advantage that it is suitable for image retrieval system to reduce quantization error as well as reducing memory and time complexity. Small quantization error is the most significant factor for image retrieval performance. Inverted file has a intrinsic characteristic that the larger codebook size it has, the smaller average length of the image file lists it has. For this reason, we can expect faster database search, as the codebook size gets larger. In short, the inverted file structure enables more accurate performance with larger size of codebook, simultaneously more efficient performance in

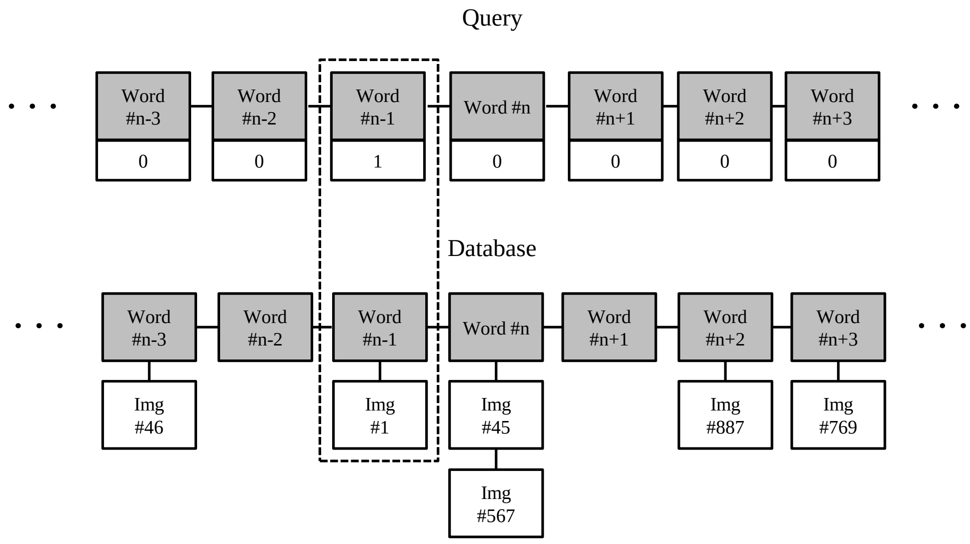


Figure 2.5: Illustration of inverted file structure. A rectangle filled with gray denotes a codeword with its index. A number in white rectangle of the query denotes how many times the codeword appears in the query image, in other words, a bin value of the query BoW histogram. A white rectangle in database denotes an image containing the codeword itself.

terms of time complexity.

2.3 Limitations of BoW Representation

In the BoW representation based image retrieval, several limitations, quantization effect caused by codebook size and lack of geometric information, arise since there are intrinsic differences between text domain and image domain. In addition, regardless of the intrinsic differences, frequently occurring codewords in many images (documents) diminish inter-image (inter-document) discriminability. Therefore, many approaches are Beside, various techniques are used to improve image retrieval performance here.

Quantization Error One major difference between text domain and image domain is that images have no word itself against text documents. In the text domain, each text word has its obvious meaning and can be described by several alphabets exactly. In addition, we know finite number of pre-defined homonyms and synonyms as well, and then we can define relationships for all finite number of words. Consequently, it is possible to distinguish whether a pair of words is same or not exactly. In case of image retrieval, however, no precise words matching is guaranteed yet because the visual words, which are forcibly generated by describing local regions and quantizing the descriptor space, are more hard to be characterized than text data. For this reason, the quantization error is one of major factor that harms the retrieval system. In Figure 2.6, local descriptors which are supposed to be assigned to the different codewords (circles and triangles) are assigned to the same codewords with the small codebook size (left) contrary to the proper codebook size (center). This kind of problem causes miss-matches, for example, a miss-match in the middle one of Figure 2.7 arises since the two different local descriptors on the right image

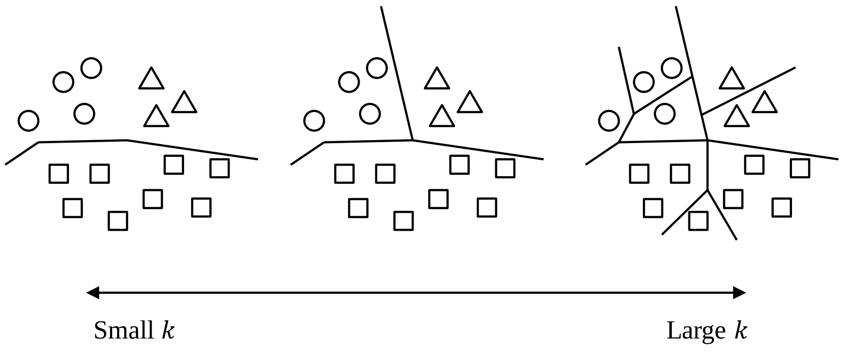


Figure 2.6: Illustration of tree local descriptor spaces with various codebook size (k). Markers of each shape (circle, triangle, square) denote noisy local descriptors, which should be assigned to a same codeword.

	Small codebook (small k)	Large codebook (large k)
Quantization error	large	small
Sensitivity to noisy descriptors	insensitive	sensitive

Table 2.2: Tradeoff between small codebook and large codebook.

are assigned to the same codeword denoted by a black circle. To overcome the problem of quantization error, we set large size of codebook as shown in the bottom one of Figure 2.7, in which the local descriptor causing a miss-match before is assigned to another codeword denoted by black star. For this reason, in general, many image retrieval system based on BoW representation uses large codebook consisting more than 1M codewords [6, 8].

Noisy Descriptors In order to reduce quantization error which is a major factor to harm retrieval performance, we set large k. However, one another problem caused by noisy descriptors arises with too large k: image retrieval system has a trade-off between large k and small k as shown in Table 2.2. On the contrary to quantization error, local descriptors that are supposed to be assigned to a same codewords are sometimes assigned to different codewords when we use large codebook. For example, in the right one of Figure 2.6, noisy local descriptors marked as circles should be assigned to a same codeword, however, they are scattered to three different codewords because of too large codebook size. To compensate this phenomenon, much effort has been put. They include conducting multiple assignment [41, 13], soft assignment [3], Hamming embedding [15, 14, 13], hierarchical scoring [6, 10] or product quantization [42, 43].

Lack of Geometric Information One another major difference between text and image is that, compared to a text document, the geometric relations of visual words are very important factors in an image to recognize what the image is about. However, a basic image retrieval system applied with text retrieval approach never considers the geometric information, and it causes a serious problem of mismatches between visual words. In order to exploit geometrical que, we encode geometrical constraint [15, 14, 10, 5, 21, 22] or conduct post-processing

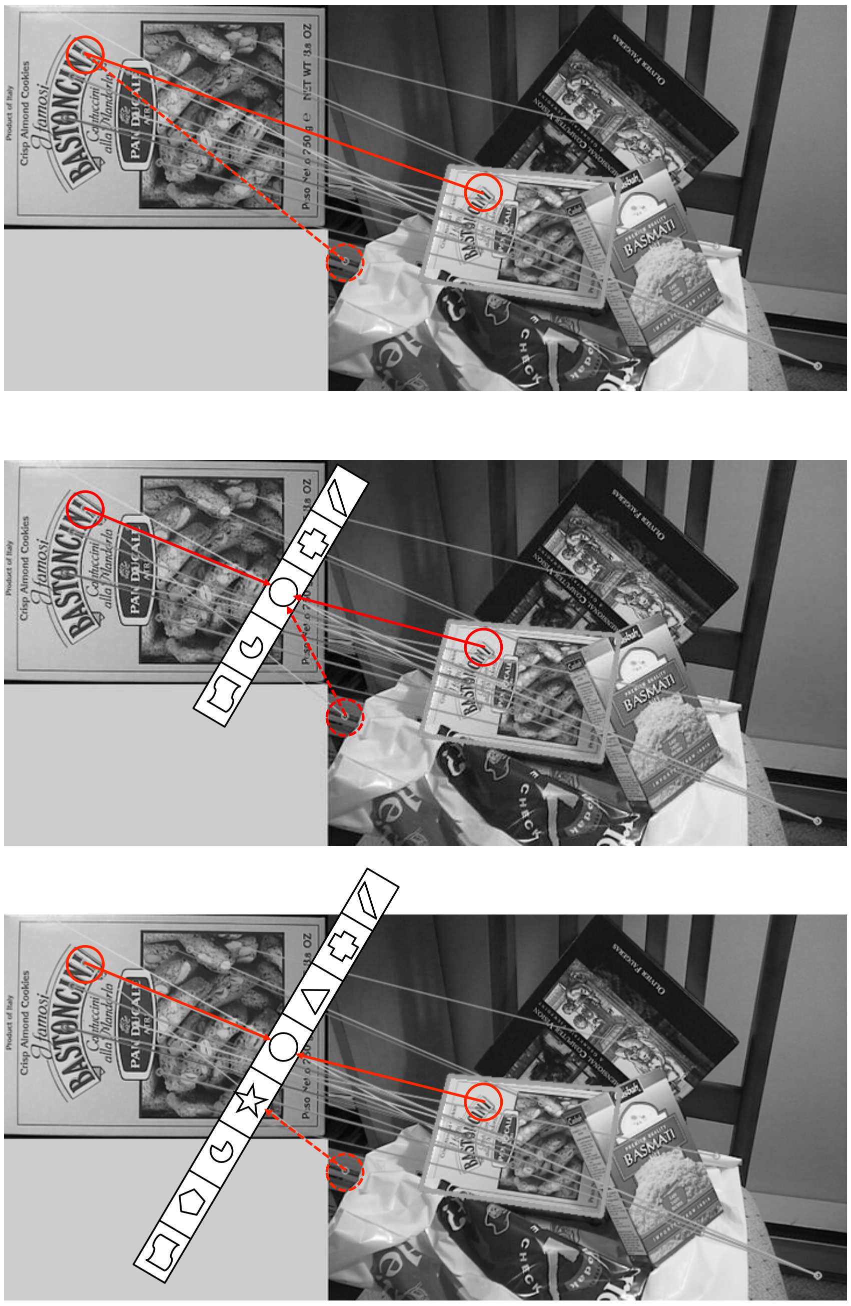


Figure 2.7: An example of image matching. The left image and the right image are matched using local descriptors. Top: The red line denotes true-match and the red dotted line denotes miss-match. Middle: Matches with small size codebook (k=5). The white array denotes codebook composed of 5 codewords. Three detected local regions are assigned to the same codeword. Bottom: Matches with large size codebook (k=8). The white array denotes codebook composed of 8 codewords. Two detected local regions, which should be matched, are assigned to the same codeword. The other detected region (red dotted circle), which should not be matched with the region (red circle) of left image, is assigned to another codeword.

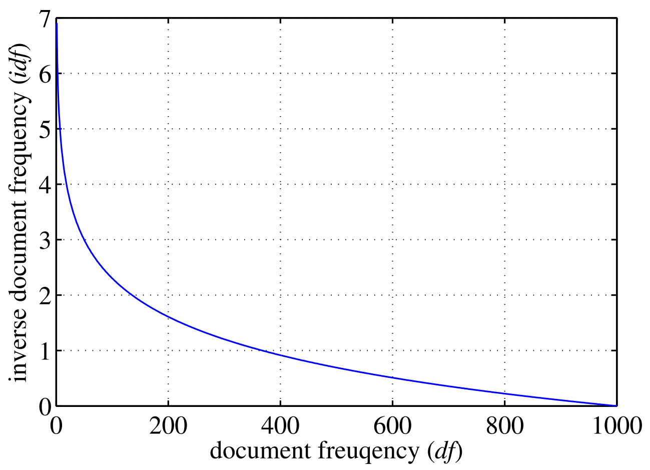


Figure 2.8: A function that converts document frequency (df) to inverse document frequency (idf) in the tf-idf weighting scheme. The function is $idf = log\left(\frac{N}{df}\right)$ where N is the total number of documents in a database. In this example, N is 1000.

that verifies geometric relations between a query and a set of retrieved images [8, 14, 19, 16, 20]. The former encode the intrinsic information of local descriptor, especially SIFT descriptor [1], like orientation (θ) and scale (σ) into BoW representation. It excludes or weight down some local descriptors with scales and orientations being against the other descriptors' tendency. The latter re-ranks a candidate set (initially retrieved images) from a query according to a strong geometrical verification step, called Random Sample Consensus (RANSAC) [51]. Through RANSAC, we can estimate a geometrical transformation between a query and a candidate image (among initially retrieved images), in other words, we can decide which descriptors are inliers (the others are outliers). According to the number of inliers with respect to a query image, we re-rank the candidate set to get higher precision.

Frequent Codewords BoW model based image retrieval system follows typical codeword weighting scheme of text retrieval system, called tf-idf weighting, because it uses text like representation. tf-idf (term frequency and inverse document frequency) weighting scheme, which weights down the negative influences of frequently occurring words in text retrieval [40], is directly applied to image domain [2] for the same reason. This weighting method has an obvious and reasonable concept that inter-class discriminative power of frequently occurring words is substantially small. Some terms like 'is', 'was', 'and', 'or', 'a' and 'the', which occur in almost all documents, harms discriminative power because its frequency is too large relative to other terms. In order to detect that kinds of frequent word, the weighting scheme computes document frequency (df). Document frequency (df_k) of a word k means the number of documents containing the word k. Because a word of large document frequency is considered unimportantly, the weighting scheme gives it a small weight value, called inverse document frequency (idf), according to the decreasing function shown on Figure 2.8. Sivic and Zisserman [2] showed that the tf-idf weighting can improve performance in image domain as well, since then it has become the most common and general weighting scheme in the BoW representation based image retrieval.

Various Approaches for BoW Representation In the field of image retrieval based on BoW representation, various approaches for improving retrieval performance has been attempted. In order to carry recall to the pinch as high as possible, a retrieval technique called query expansion [20, 9, 23] was proposed. The idea of the query expansion is that retrieve again with only inliers, which are selected as a result of RANSAC based spatial verification, not all visual words in query. After removing outliers from BoW histograms of the query and spatially verified images, we retrieve again from database with an averaged BoW histogram as a new query. Recently, as the size of database is growing extremely because of tremendous amount of web image data, we call it large-scale image retrieval, several approaches to compactly represent BoW model has been attempted as well [52, 53]. p¿

Chapter 3. Motivation and Problem Definition

In this chapter, motivated by of real experimented data from several related works, we define problems what we willing to solve in this thesis. In Section 3.1, we introduce experiment results we motivated, and in Section 3.2 we define what we will solve.

3.1 Motivation

In this thesis, we focus on the limitations of conventional codebook weighting scheme, tf-*idf* [2], and hierarchical scoring using hierarchical codebook, called Vocabulary Tree [6]. In addition to that, we address a new problem that, so far, there has been no approaches to consider intra-class key word.

tf-idf Weighting Even though tf-*idf* weighting scheme becomes the most common framework in BoW representation based image retrieval, strangely, performance improvement by the tf-*idf* weighting is usually very low. Moreover, sometimes it rather hurts performance in some specific classes as shown in Table 3.1. Performance improvement on Oxford5k dataset after tf-*idf* weighting is only 0.47% in average. In the case of Bodleian class, it rather hurts performance by 2.67%. As the experiment shows, actually, tf-*idf* weighting gives very slight and in consistent improvement. There has been several works addressing the problem of tf-*idf* weighting. Jegou et al. [13] analyzes the limitation of tf-*idf* weighting attributes to the gap between the text domain and image domain. Text has discrete spaces and a finite alphabet. They insist, that the images have a continuous feature space so the quality of matches (the closeness of the descriptors in feature space) is important, while tf-*idf* weighting does not take this into account. They tried to overcome the limitation of tf-*idf* weighting by additionally using quality of matches (Hamming distances) with the conventional tf-*idf* weighting. Wang el al. [10] also tried to overcome the limitation by additionally using the path information of which the descriptor has followed in hierarchical codebook (Vocabulary Tree [6]) along with tf-*idf* weighting.

Hierarchical Scoring Nister and Stewenius [6] generated very large codebook efficiently through hierarchical k-means clustering in order to reduce quantization error and secure discriminative power. We call such kind of codebook hierarchical codebook because it has a hierarchical structure as shown in Figure 3.1. In the toy example of Figure 3.1, which has branch factor of 2 and depth of 3, the nodes are divided into two clusters with k-means clustering at the first level. The two clusters are then divided into two clusters respectively, and the nodes in the lower levels go through the same process until they reach the pre-defined depth. In the end, we earn very huge codewords within a feasible time. Although it can bring about cumulative miss-quantization as it goes to a larger level, it has a powerful advantage of hierarchical scoring as well as providing a very large codebook, more than 1M. Hierarchical codebook based image retrieval uses hierarchical scoring which treat not only leaf nodes but also

	Average Precision(%)		
Class	Uniform weight	*idf* weight	Improvement
All souls	49.31	50.47	+1.17
Ashmolean	55.61	57.92	+2.31
Bodleian	55.73	53.06	**-2.67**
Christ church	58.85	59.14	+0.29
Hertford	70.53	71.09	+0.56
Magdalen	12.22	13.27	+1.05
Radcliffe camera	65.42	66.01	+0.59
mAP(%)	52.53	53.01	+0.47

Table 3.1: Experiment by Cai et al. [11] representing performance improvement after tf-idf weighting on Oxford5K dataset [8]. mAP denotes a mean value of average precisions (AP) of each class.

Levels considered	UKbench (Top-4)	Oxford5K (mAP)
6	3.16	43.9
5, 6	**3.07**	**41.8**
4, 5, 6	3.29	**37.2**
3, 4, 5, 6	3.29	**35.3**

Table 3.2: Retrieval performance with various number of levels considered on UKbench dataset by Nister and Stewenius [6] and Oxford5K dataset by Phibin et al. [8]. Top-4 score of UKbench dataset, a specified performance metric for the dataset, denotes average number of relevant images for each class.

nodes of different levels as codewords to compensate problems like being sensitive to noisy descriptors. In other words, we reduce quantization error by considering leaf nodes as codewords, at the same time, we compensate negative effect caused by noisy descriptors by considering parent nodes as codewords. For this reason, generally, the performance becomes better if proper multiple levels are considered in hierarchical scoring. On the other hand, however, hierarchical scoring sometimes lowers the precision as shown in Table 3.1. In the result [6] on UKbench dataset, hierarchical scoring with two levels (level 5, level 6) shows lower performance than flat scoring (level 6) although the performance usually improves when we consider more numbers of levels. Oxford5k dataset results [8] shows the problem more apparently. It shows that using more numbers of levels degrades the performance. Phibin et al. [8] analyzed that data points may be suffering from bad initial point in the level closer to the root, which is denoted by cumulative miss-quantization in this thesis. They added that hierarchical scoring has not yet overcome this problem and that more work is needed to understand the phenomena. Likewise, hierarchical scoring has low consistency in the number of levels that are considered in scoring, the improvement and the dataset.

Intra-class key-word When we focus on codewords weighting methods of image retrieval based on very large dimensional BoW representation, most of the methods that have been studied to increase the distinction between classes, do not take the fact that images of different classes have different key words into consideration. Only

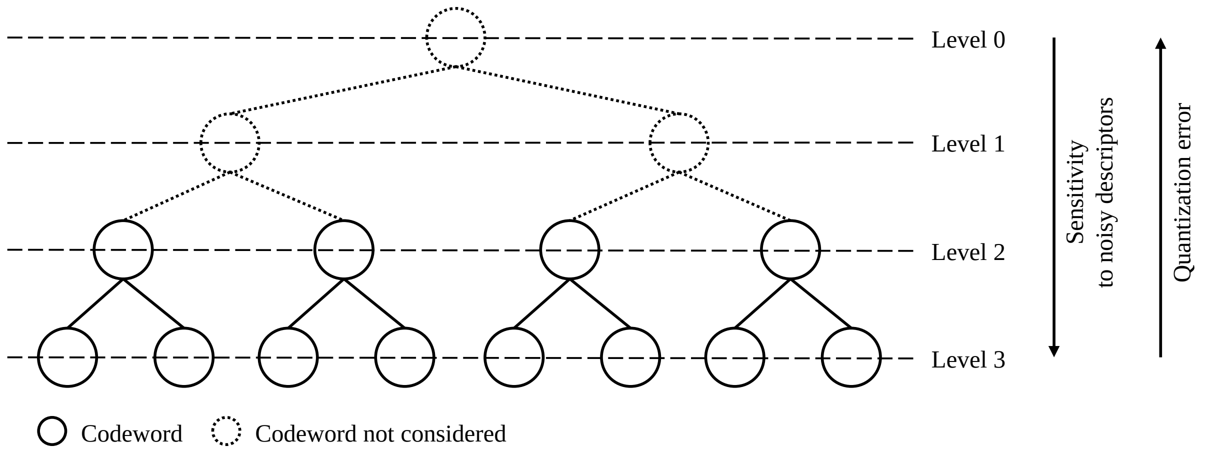


Figure 3.1: Illustrations of hierarchical codebook [6] with branch factor of 2 and depth of 3. In this case, the number of levels considered in hierarchical scoring is 2 (from depth 2 to 3).

same standard is applied to the images of every class, in other words, every class images applies a same weight vector independent with what their class is. tf-idf weighting method [2], most retrieval systems use, generate a single weight vector obtained on a given database. Image retrieval systems of [5, 7] select most distinctive codewords for every class using boosting algorithm [4]. Weighting methods for encoding spatial information [15, 14, 10, 5, 21, 22] into BoW representation apply different weights to codewords according to geometrical relations in local descriptor level, not in class level. Weighting methods which consider quality of match, called multiple assignment [41, 13], soft assignment [3], Hamming embedding [15, 14, 13], hierarchical scoring [6, 10] or product quantization [42, 43], also apply different weights to codewords according to distances between a codeword and local descriptors in local descriptor level, not in class level. There are two reasons why the codewords weighting method considering intra-class key words has not been attempted. The first reason is that recent BoW representation has too large dimensionality because of too large image database. Therefore, existing machine learning method cannot be applied to image retrieval, which is not free from time and computational complexity. Secondly, as shown in Table 1.1, the real scenario of image retrieval does not have finite number of classes and the database can not actually be categorized. However, if we assume a retrieval system on an image database pre-categorized, a weighting method considers intra-class key words is needed because every class has their key words which represent the class most well. For example, in text domain, words like 'gene' and 'inheritance' are important in the document class of genetics, on the other way, words like 'gravity' and 'velocity' are important in the document class of physics. In image domain, as shown in Figure 3.2, codeswords about stripe patterns are key words in zebra class, on the other way, codewords like dots patterns are key words in leopard class.

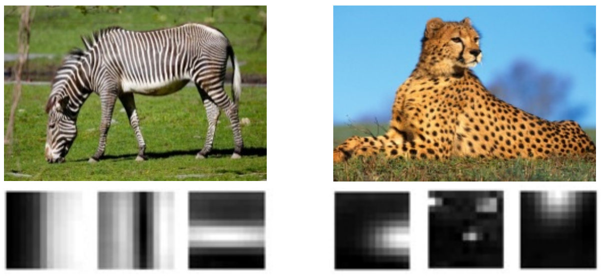


Figure 3.2: Illustrations of intra-class key words (important codewords) in zebra class (left) and leopard class (right).

3.2 Problem Definition

tf-idf weighting and hierarchical scoring which have been widely used in image retrieval based on very high dimensional BoW representation. However, it not only brings too slight improvement and inconsistent result according to different datasets but also rather degrades the performance as shown in Table 3.1, 3.2. To solve this problem, we suggest database specific weighting strategies of *tf-idf* weighting and hierarchical scoring that can bring the optimal result to the given objective database. The proposed method maintains the frames of conventional *tf-idf* weighting and hierarchical scoring without any modification. Moreover, we do not consider extra cue like geometric information, quality of the descriptor match, or path information of the descriptors in hierarchical codebook. We focus on improving *tf-idf* weighting and hierarchical scoring themselves through learning procedure that only takes document frequency (*df*) and level (l) of hierarchical scoring information into account. In addition to that, through the proposed database specific weighting strategies, we considers intra-class key words to differently give relevant weight values to codewords for different classes. In conclusion, we propose a codewords weighting method, which not only solves the problem of *tf-idf* weighting and hierarchical scoring but also make intra-class key words into consideration, for the image retrieval based on very high dimensional BoW representation.

Chapter 4. Learning Weight Mapping Function

The proposed codebook weighting method applies relevant weight to each codeword, following the conventional assumption of BoW representation that each codeword occurs independently in an image [13]. We define several types of signature that is closely related to inter-class discriminability and calculate the signature value of each codeword. In other words, a codeword has its own signature value and that is correlated with importance of the codeword. Then, a relevant weight value is assigned to each codeword, according to its signature value. For example, if a signature type is in inverse proportion to importance, a codeword with a large signature value gets a small weight value. In a similar way, a codeword with a small signature value gets a large weight value. We denote the signature value obtained from each codeword as key-signature and the function which assigns weight from the key-signature as Weight Mapping Function (WMF).

The overall framework of the proposed method is described in Figure 4.1. We randomly sample a number of pre-labeled classes from the given database and use them as training data. In actual scenario, we label a pre-defined number of sample images of given database and use them as training data. After that, we compute key-signature values of pre-trained codewords using the training data. We then train WMF that can maximize the inter-class distinction of the sampled training data. From the learned WMF, we can convert the key-signature value of a codeword into a weight value and eventually obtain the weight vector that corresponds to one codebook. In query time, we apply the weight vector to the BoW histogram of query and that of database, and score the results. That is, our method makes additional memory usage and additional computational cost almost zero in on line stage, because what we only have to do is applying the weight vector learned in the off-line phase to the scoring function in on-line phase. Note that we do not consider any other extra queues like quality of matches, geometric information, path of a descriptor goes down in hierarchical codebook, et cetera.

We will define three types of WMF in Section 4.1 and three types of key-signature in Section 4.2. After that, objective function to learn WMF using defined key-signatures and its optimization method are described in Section 4.3 and Section 4.4 receptively.

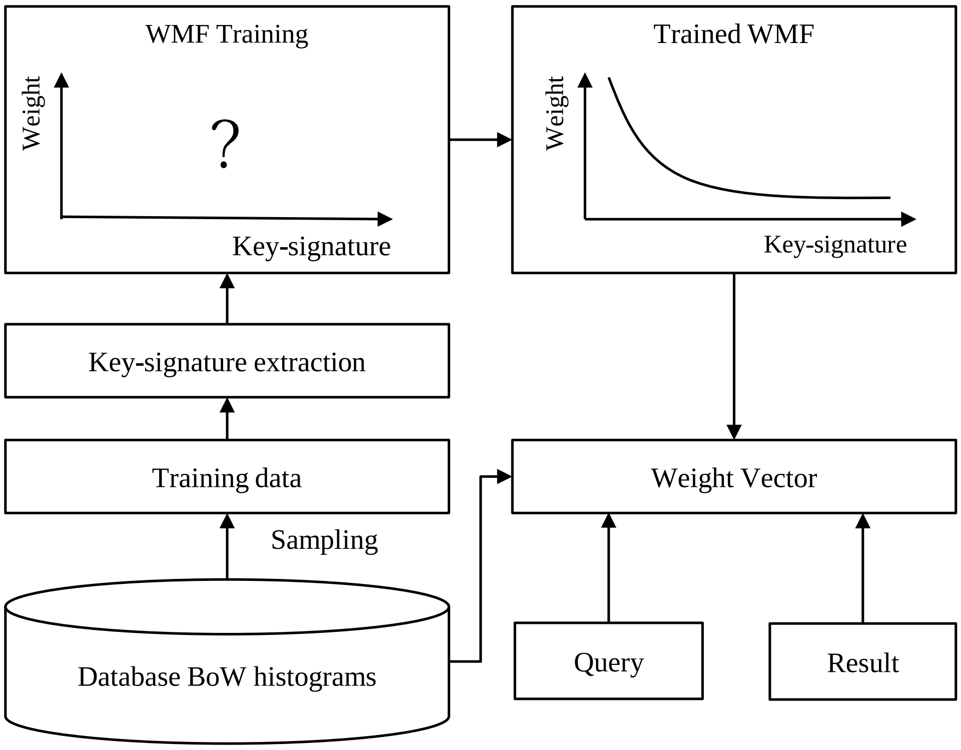


Figure 4.1: Flow chart of the proposed system. We train Weight Mapping Function (WMF) in off-line phase, which converts a key-signature value for each codeword to a weight value, in order to get a weight vector. The trained weight vector is used when we score up database vectors from a query vector in on-line phase.

4.1 Weight Mapping Function (WMF)

Weight Mapping Function maps a key-signature value of each codeword into a weight value:

$$w_k = WMF(s_k), 1 \leq k \leq K, \tag{4.1}$$

where s_k is the key-signature value of k-th codeword of K size codebook and w_k is the weight value mapped by weight mapping function $WMF(\cdot)$. The expected shap of a WMF depends on the type of key-signature. If a key-signature value of a codeword is in proportion to importance of the codeword, the WMF should has increasing shape. In contrast to that, if a key-signature of a codeword is in inverse proportion to importance of the codeword, the WMF should has decreasing shape. For some types of key-signature, some codewords with moderate key-signature values can be more important than the other codewords with small or large key-signature values. In this case, the proper WMF has bell shape like Gaussian function. Therefore, appropriate freedom is needed for WMF to have various shapes. We define three types of WMF with different basis as described in Table 4.1 to experiment various shapes of WMF.

Exponential WMF with four parameters has either increasing or decreasing shape and n-th degree polynomial WMF with $(n + 1)$ parameters has $(n - 1)$ number of extremums at most. Exponential WMF, of course, cannot have extremums, on the other hand, polynomial WMF can have. Moreover, we defined a special type of WMF that

WMF Type	WMF	Shape
Exponential	$WMF_{exp}(s_k) = p_1 \cdot exp(p_2 \cdot s_k + p_3) + p_4$	Only increasing or only decreasing
Polynomial	$WMF_{poly}(s_k) = \sum_{i=0}^{n} p_i \cdot s_k^i$	Maximally $n-1$ extremums
Lookup Table	$WMF_{LUT}(s_k) = p_j, i = \{i \mid 1 \leq j \leq J\}$, where J is a possible number of the key-signature values s_k.	Free

Table 4.1: 3 different types of WMF and its possible shapes.

has a lookup table which returns a weight value of every key-signature to make WMF completely free from shape. In this case, the number J of WMF parameters is equal to the number of possible values which key-signature s_k can have.

4.2 Key-signature

To learn WMF, we have to define a key-signature under the conditions that 1) it can be independently computed from each codeword and 2) it should related to the discriminability. That is, the key-signature can be regarded as the importance factor of the codeword. If a type of key-signature is highly correlated with discriminative power, the shape of leaned WMF would show more clear tendency, as a result, we can expect better retrieval performance. It is the reason why defining key-signature is the most important part in the proposed method. In this thesis, we defined three types of key-signature: document frequency, level of hierarchical codebook and intra-class scale-normalized standard deviation. As shown in Table 4.2, document frequency and level are computed from whole database and converted to a weight value shared by every class. However, in order to consider intra-class key words, intra-class scale-normalized standard deviation is computed from a class data and converted to a weight value for a class only. In other words, with respect to a codebook, a vector of document frequency or level is converted to a single global weight vector, on the contrary, a vector of intra-class scale-normalized standard deviation is converted to a intra-class weight vector.

In this section, we define the three types of key-signature, Document frequency, level and intra-class scale-normalized standard deviation, in sub-section 4.2.1, 4.2.2 and 4.2.3 respectively.

4.2.1 Document Frequency

In conventional tf-idf weighing scheme, term frequency tf_k means the number of k-th codeword contained in an image. Here we can consider tf_k as a value of k-th bin of a BoW histogram. Document frequency df_k of a k-th codeword means the number of images contain the codeword in a database, and inverse document frequency idf_k is the logarithmic value of the reciprocal of df_k like:

$$idf_k = log\left(\frac{N}{df_k}\right), \tag{4.2}$$

Key-signature type	Computed using	Converted to (by WMF)
Document frequency (df_k)	Whole database	Global weight value (w_k)
Level (l_k)	Whole database	Global weight value (w_k)
Scale-normalized standard deviation ($\sigma^{SN}_{y,k}$)	y-th class data	Intra-class weight value for y-th class ($w_{y,k}$)

Table 4.2: Comparison of three different types of key-signatures of k-th codeword. y denotes a class index.

where N is total number of images in database. In tf-idf weighting scheme, idf_k plays a role as an weight value of the k-th codeword because it is multiplied by tf_k when we compute a distance between a query image and a database image. As a k-th codeword frequently occurs in many images, df_k gets larger, and then idf_k becomes smaller. The more times the codeword occurs in many images, the less important the codeword becomes. Since a codeword has been considered to have a larger discriminative power with smaller df_k, we can think df_k is correlated with importance of the codeword. Therefore, we define df_k as a key-signature of the k-th codeword. In this case, the WMF of the original tf-idf weighting scheme is

$$w_k = WMF(df_k) = log\left(\frac{N}{df_k}\right) = log(N) - log(df_k), \tag{4.3}$$

where w_k is the mapped weight value of k-th codeword by the $WMF(\cdot)$. Since $WMF(\cdot)$ is a decreasing function in original tf-idf weighting as shown in (4.3), codewords with larger df_k are assigned smaller weight (idf_k). Our goal here is to find the most relevant WMF in the database we are trying to retrieve.

4.2.2 Level of a hierarchical codebook

Level of a hierarchical codebook is described in Figure 3.1. Codewords on a level closer to leaf (level 3 in Figure 3.1) have small quantization error because the descriptor space finely splits into much more cells. Therefore, we can think codewords on a level closer to leaf level have better discriminability. However, in terms of sensitivity to noisy descriptors, Codewords on a level closer to root (level 0 in Figure 3.1) are robust to noisy descriptors. For this reason that hierarchical codebook shows trade-off between higher level and lower level, proper balance between them is needed to enhance discriminative power. In order to find optimal weights according to levels, we define the level of hierarchical codebook as another type of key-signature. The conventional hierarchical scoring does not apply the different weight to different levels, but equally treats the codewords that belong to the same level, like

$$w_k = WMF(l_k) = 1, \tag{4.4}$$

where l_k is the level and w_k is the mapped weight value of k-th codeword by the weight mapping function $WMF(\cdot)$. In the conventional hierarchical scoring, a drawback that the quantization error significantly increases around the root was overcome by tf-idf weighting. Our goal here is to find the most relevant WMF for the database we are trying to retrieve and apply it to hierarchical scoring, only using the level.

4.2.3 Intra-Class Scale-Normalized Standard Deviation

For a class, intra-class standard deviation (ICSTD) of a codeword frequency can be a nice intra-class key-signature because we can consider a codeword with a small ICSTD of its frequencies as a key-word in the class. On the other hand, a codeword with large ICSTD can be considered unimportantly in the class because some images of the class may have large number of this codeword, however, the other images may not. In order to use this kind of value as a key-signature, we have to find the standard deviation of each codeword at each class. However, getting the standard deviation without any normalization causes a problem of fairness because the scales of a BoW histogram bin in a class vary a lot. In other words, an important codeword in a class can have large standard deviation of its frequency because it occurs very frequently in images of the class. In Figure 4.2(a), we made four BoW histograms with a hierarchical codebook [6] of 1M size (we visualize only 50 codewords among 1M codewords) and expressed the standard deviation of each codeword frequency. Here we can confirm that the scales of codeword frequencies are varying with significant differences. If we just use the standard deviation as key-signature at this stage, it becomes inequitable because most of the codewords with large frequencies possess large standard deviations compared to the other codewords with small values. To solve such problem, scale of each codeword frequency (each histogram bin) should be normalized like

$$\mathbf{d}_{y,z}^{SN} = \left[d_{y,z,1}^{sn}, \cdots, d_{y,z,k}^{sn}, \cdots, d_{y,z,k}^{sn}\right]^\top, \quad d_{y,z,k}^{sn} = \frac{d_{c,z,k}}{max(d_{y,1,k}, \cdots, d_{y,z,k}, \cdots, d_{y,Z_y,k})} \tag{4.5}$$

where $\mathbf{d}_{y,z}^{SN} \in \Re^K$($K$ is codebook size) is a scale normalized BoW histogram after L_1-normalization of image $z \in \{z|1 \leq z \leq Z_y\}$ of class y and Z_y is the number of images of class y. $d_{c,z,k}$ denotes a k-th dimension value ($k \in \{k|1 \leq k \leq K\}$) of a image z of class y and $d_{y,z,k}^{sn}$ denotes a scale normalized value of $d_{c,z,k}$. The function $max(.)$ returns a maximum value among the values in the bracket. In Figure 4.2(b), dotted lines indicate the values which have been obtained through scale normalization of each BoW histogram bin with (4.5) and their standard deviations (red line), called intra-class scale normalized standard deviation (intra-class SNSTD). Comparing the intra-class SNSTD (Figure 4.2(b)) with the original one (Figure 4.2(a)), the intra-class SNSTDs equitable to codewords of every scale has been obtained.

In order to verify SNSTD, the proposed key-signature, we conducted a toy experiment which shows that 1) we can index codewords in order of importance using SNSTD and 2) important codewords in a class are different from other class. We designed the toy example under the assumption that a variance of frequencies of a codeword within a class could be used to determine the importance of the codeword. It was conducted with a simple dataset we made as shown in Figure 4.3. We made the small dataset about fruits composed of banana, strawberry, kiwi and blueberry classes because these classes have their representative color. We then clustered RGB color vectors of each pixel of every image to represent an images with a bag of color words histogram like [33, 34]. The clustered vectors were then made into 30 color codewords through k-means clustered algorithm. After that, we described each image with a histogram using 30 color words and sorted the color codewords in the direction where the SNSTD (4.5) increases because the variance of a codeword is expected to be in inverse proportion to its importance. Figure 4.4 shows the experimental results of sorting the color codewords of images in each class

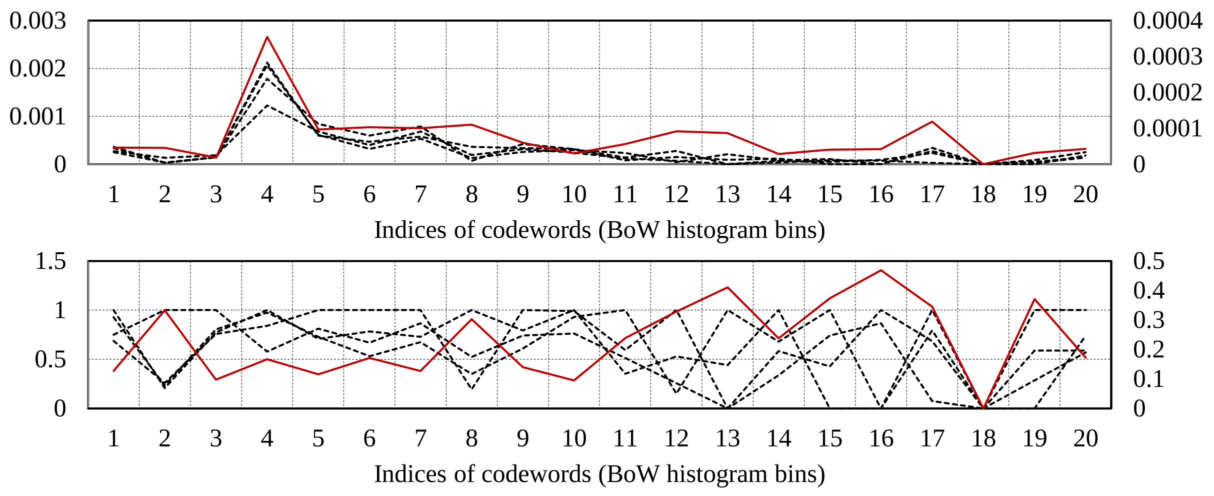


Figure 4.2: Among 1M codewords made up by using hierarchical clustering [6] with a depth of 6 and branch factor of 10, we used the first top 50 codewords to visualize BoW histograms and marked the values with a line in order to be readily recognized. (a): L_1-normalized BoW histograms of four images (black dotted line) in a class and standard deviations for each histogram bins (red line). (b): Scale normalized BoW histograms (black dotted line) of (a) and its standard deviations (red line).

by the SNSTD. Reddish codewords with SNSTD of 0 lie above yellowish codewords in the sorted result of the banana class. The reason why the SNSTD of these reddish colors is 0 is that the images of the banana class do not include a single reddish codeword. Even though the reddish color words are not used to express the banana class, they can play a role as a key-word, since they should not appear in the banana class. Right next to the reddish codewords are yellowish codewords which actually best express the banana class. These yellowish codewords should be key words of the banana class because it is the most representative color. The following color words after the yellowish codewords are less important ones with high standard deviations, only frequently appearing in some images of the banana class or rarely appearing in the other images. Similarly, the rest of the classes show the same result. In conclusion, we investigated the validity of SNSTD as a key-signature with a simple experiment and the result shows SNSTD can be used as a key-signature type because it shows strong tendency of inverse proportion to codeword importance.

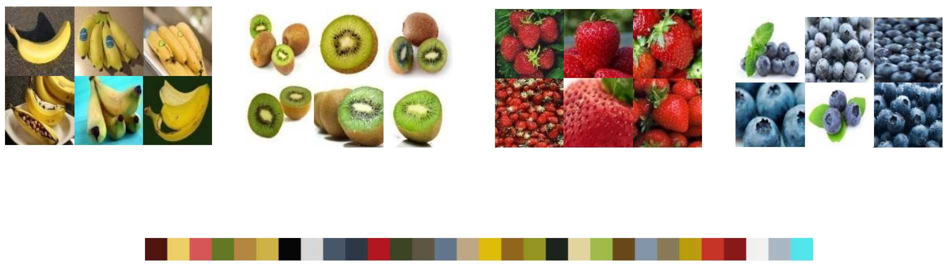


Figure 4.3: Top: Small database containing 24 images of 4 fruit classes obtained by randomly selecting the result of Google image search from text queries. From left to right, the image class is banana, kiwi, strawberry and blueberry. Bottom: A codebook obtained from the database of Figure 4.3. Using every RGB vectors of uniform grid points as local descriptors, the codebook is generated by k-means clustering with $k = 30$.

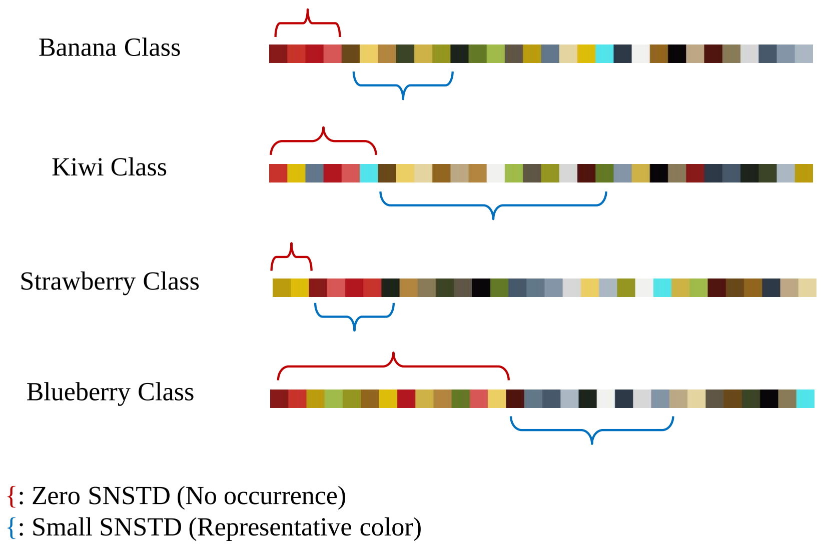

Banana Class
Kiwi Class
Strawberry Class
Blueberry Class
{: Zero SNSTD (No occurrence)
{: Small SNSTD (Representative color)

Figure 4.4: Sorted color codewords in increasing order of SNSTD computed from (4.5) in each class.

4.3 Cost Function

We have to learn the parameters of WMF described in Table 4.1 that maximize the discriminability among the training class by using the randomly sampled training data. We randomly sampled M training classes from a database $D = \{d_n | 1 \leq n \leq N\}$ which has total N BoW histogram d_ns, and then made a training set $D^{tr} = \{d_n | 1 \leq n \leq N^{tr}\} \subset D$ which has N^{tr} number of entities of BoW histogram d_ns. Each training class of class id $m (1 \leq m \leq M)$ has G^m number of entities of BoW histograms. We will call a set of BoW histogram ids that a single training class m has as $I^m = \{i_g^m | 1 \leq g \leq G^m)$.

Our goal here is to design a cost function,

$$\hat{\mathbf{p}} = \arg \underset{\mathbf{p}}{min}\, cost(\mathbf{p}), \tag{4.6}$$

about the set of parameters of WMF $\mathbf{p}$, optimize it, and find the $\mathbf{p}$ that can maximize the discriminability. The definition of the parameter set $\mathbf{p}$ is described in Table 4.1. The proposed cost function is composed of three terms,

$$\begin{aligned} cost(\mathbf{p}) &= -T_{mAP}(\mathbf{w}) + \lambda_1 \cdot T_{guide}(\mathbf{w}) + \lambda_2 \cdot T_{regularization}(\mathbf{w}), \\ \mathbf{w} &= \begin{bmatrix} w_1, & w_2, & \cdots, & w_K \end{bmatrix}^\top, \ w_k = WMF^{\mathbf{p}}(s_k) \end{aligned} \tag{4.7}$$

with λ_1 and λ_2 that balance the ratio of three terms. $WMF^{\mathbf{p}}(\cdot)$ means a WMF which has parameter $\mathbf{p}$. The first term $T_{mAP}(\mathbf{w})$ is the mean average precision (mAP) within the training data. To make smaller cost value with larger mAP, we multiply -1 with $T_{mAP}(\mathbf{w})$ in (4.7). The second term $T_{guide}(\mathbf{w})$ works as a guide for a optimizer to go better way that decreases the first term (increases mAP). Finally, $T_{guide}(\mathbf{r})$ is a regularization term, in order to avoid over-fitting, that makes the curve of WMF smooth. The following is detailed explanation of these three terms.

mAP Term The term $T_{mAP}(\mathbf{w})$ calculates mAP that does not interpolated. mAP is a mean value of average precisions with respect to every queries. Here, the average precision (AP) means that average value of precisions of all ground truth images with respect to a query. mAP, same as the term $T_{mAP}(\mathbf{w})$, is computed like

$$T_{mAP}(\mathbf{w}) = mAP(\mathbf{w}) = \sum_{m=1}^{M} \sum_{g=1}^{G_m} \frac{g}{r(\mathbf{q}_w^m, \mathbf{d}_w^{i_g^m})}, \tag{4.8}$$

for a test query set $\{\mathbf{q}^m | 1 \leq m \leq M\} \subset D^{tr}$ in training data D^{tr} after applying a weight vector $\mathbf{w}$. The subscript w means that a vector $\mathbf{x} \in \Re^K$ is weighted by $\mathbf{w}$ like

$$\mathbf{x}_w = \begin{bmatrix} w_1 \cdot x_1, & w_2 \cdot x_2, & \cdots, & w_K \cdot x_K \end{bmatrix}^\top \tag{4.9}$$

and $r(\mathbf{q}_w^m, \mathbf{d}_w^{i_g^m})$ returns the ranking of a vector $\mathbf{d}_w^{i_g^m}$ (g-th vector in class m which has a vector id i_g^m) with respect to the m-th class query $\mathbf{q}_w^m$. Finally, we multiplied $T_{mAP}(\mathbf{w})$ by -1 as shown in (4.7) to maximize mAP of training data when we minimize the cost.

Guide Term However, as expected in equation (4.8), in spite of variation of $\mathbf{w}$, mAP does not change if the order of vectors in Figure 4.5 does not changes with respect to queries. For this reason, the optimization ends up without significant effect if we only use $T_{mAP}(\mathbf{w})$ as a cost function. We therefore added a term $T_{guide}(\mathbf{w})$ to make the cost continuously change, and simultaneously, induce mAP to increase with the variation of $\mathbf{w}$:

$$T_{guide}(\mathbf{w}) = \sum_{m=1}^{M} \sum_{g=1}^{G_m} \frac{\left\| \mathbf{q}_w^m - \mathbf{d}_w^{i_g^m} \right\|_1}{\sum_{n=1}^{N^{tr}} \|\mathbf{q}_w^m - \mathbf{d}_w^n\|_1 \cdot B\left(\mathbf{q}_w^m, \mathbf{d}_w^{i_g^m}, \mathbf{d}_w^n\right)}, \tag{4.10}$$

where $\|\cdot\|_1$ means L_1-norm and $B(\cdot)$ is an binary indicator function,

$$B\left(\mathbf{q}, \mathbf{d}^1, \mathbf{d}^2\right) = U\left(r(\mathbf{q}, \mathbf{d}^1) - r(\mathbf{q}, \mathbf{d}^2)\right), \tag{4.11}$$

which returns 1 if $\mathbf{d}^2$ ranks higher than $\mathbf{d}^1$ with respect to $\mathbf{q}$ and 0 for otherwise since $U(\cdot)$ is an unit step function that returns 1 for a input of a non-negative value and 0 for otherwise. Therefore, the denominator of (4.10) denotes

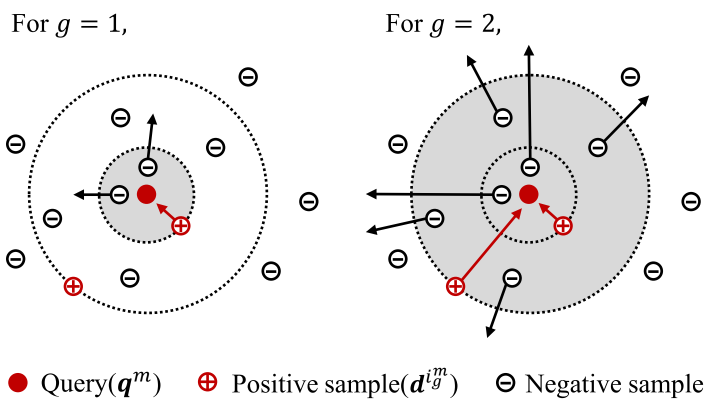


Figure 4.5: Illustration of the role of the guide term $T_{guide}(\mathbf{w})$ in equation (4.10). A Red circle means the query $\mathbf{q}^m$ of class m, circles with pluses inside denote positive vectors (ground truth vectors) of class m with respect to the query $\mathbf{q}^m$. Circles with minuses inside denote negative vectors which are not in class m. The image on the left shows the first positive vectors (first ground truth vector) ranked at 3rd pushing out the negative vector and approaching closer to the query by the guide term $T_{guide}(\mathbf{w})$. Likewise, the image on the right is about the second positive vector ranked at 8th.

the sum of L_1-distances between negative vectors, that are ranked higher than positive vector $\mathbf{d}_w^{i_g^m}$, and query $\mathbf{q}_w^m$. The numerator, on the contrary, is sum of the L_1-distances between positive vectors $\mathbf{d}_w^{i_g^m}$ and query $\mathbf{q}_w^m$.

Figure 4.5 shows the role of the equation (4.10). For (4.10) to get smaller, positive vectors should reach closer to the query according to the decreasing numerator in (4.10). On the contrary to this, the negative vectors ranked higher than the positive vectors should go farther from the query according to the denominator in (4.10). Figure 4.6 is real data which show the correlation between the $T_{guide}(\mathbf{w})$ value and mAP with respect to 300 cases of $\mathbf{w}$. The correlation coefficient value between $T_{guide}(\mathbf{w})$ and mAP is -0.97, so it shows that $T_{guide}(\mathbf{w})$ works well as a guide for an optimizer to increase mAP.

Regularization Term Finally, to prevent WMF from over-fitting on the training set, especially $WMF_{LUT}(s_k)$, we added a term $T_{regularization}(\mathbf{w})$ which gives penalty to the WMF with severely fluctuating shape:

$$T_{regularization}(\mathbf{w}) = \int_{s_{min}}^{s_{max}} \left| \frac{\partial^2}{\partial s^2} WMF(s) \right| \cdot ds \tag{4.12}$$

$T_{regularization}(\mathbf{w})$ refers to the area surrounded by second derivative of WMF and s-axis. The more drastically the fluctuation repeats, the larger the $T_{regularization}(\mathbf{w})$ becomes. Discrete version of (4.12) was used to calculate the value. Figure 4.7 shows the effect of the regularization term on Oxford5K dataset. The fluctuated WMF without the term (4.12) is smoothed like blue line with the term added. If we use the term (4.12) with proper λ_1 in (4.7, we can secure generallity, which improves performance on test data as well as training data.

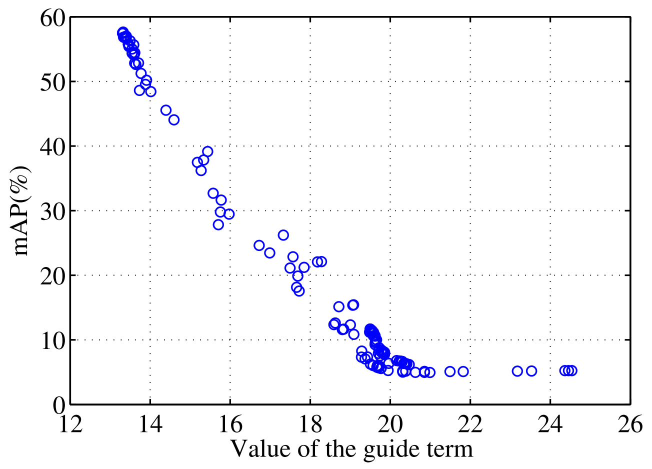


Figure 4.6: Plot of mAP(%) and $T_{guide}(\mathbf{w})$ on Oxfored5K dataset. Each point is generated from identical weight vectors. The correlation coefficient of mAP(%) and $T_{guide}(\mathbf{w})$ is -0.97.

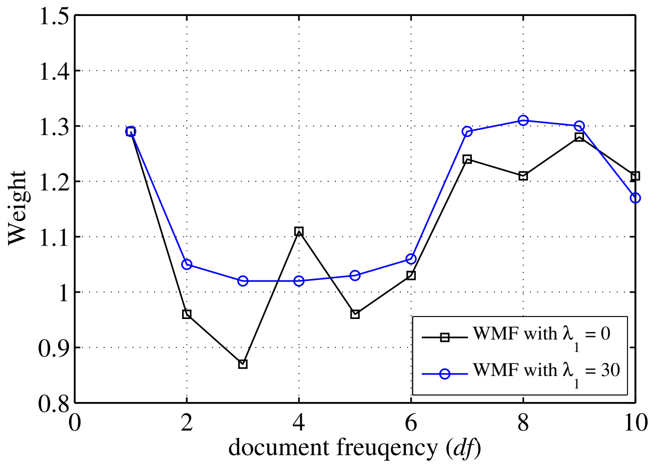


Figure 4.7: Plot of two WMFs of LUT type with document frequency as a key-signature. The WMFs were learnt on Oxford5K dataset. λ_1 denotes the balancing factor in (4.7)

4.4 Optimization

When we are about to optimize the cost function (4.7) to get optimal weight vector $\mathbf{w}$, a problem of too high computational complexity arises. Before optimization, we reduced computational complexity by means of preparing pre-computed data which will be directly used for each iteration of optimization.

Computational Complexity Reduction The parts with the largest computational cost in our cost function are 1) that we have to find the weighted L_1-distance between N^{tr} number of entities of K-dimensional training data and M number of entities of queries every-time in $T_{guide}(\mathbf{w})$ ($M \times N_{tr} \times K$ vector element wise computation is required.) and 2) that there are too many parameters in case of $WMF_{LUT}(s_k)$. In order to reduce the first one, we apply two techniques: pre-calculating distance vectors and dimensionality reduction via key-signature quantization. We pre-calculated $M \times N^{tr}$ distance vectors ($\mathbf{v}_{m,n}$) of M queries to all N^{tr} training images like

$$\mathbf{v}_{m,n} = \mathbf{q}_m - \mathbf{d}_n, \tag{4.13}$$

where $m \in \{m|1 \leq m \leq M\}$ is a class id and $n \in \{n|1 \leq n \leq N^{tr}\}$ is a training image index. However, because the dimensionality of the distance vector 4.13 is too large (e.g. K=1M), it is needed to reduce the dimensionality if possible. In order to reduce dimensionality, as shown in Figure 4.8, after we extracted a key-signature vector $\mathbf{s}$ from a codebook, we quantized the key-signature elements $\{s_k|1 \in k \in K\}$ into key-signature cluster centroids $\{s_h^{qtz}|1 \in h \in H \ and \ H \ll K\}$ by means of 1-dimensional k-means clustering. And then, we compressed a distance vectors by summing its elements values of each same cluster as shown in Figure 4.9. The second complexity that too many parameters in $WMF_{LUT}(s_k)$ was resolved by the key-signature quantization simultaneously because the number of possible key-signature values was reduced to the number of clusters at most (from J to H, $H < J$). In other words, the number of possible parameters of $WMF_{LUT}(s_k)$ was diminished to H at most.

Optimization Strategy In order to minimize the cost function and find optimal shape parameters $\hat{\mathbf{p}}$ (Table 4.1) of WMF, we utilized MATLAB built-in functions both ”fmincon” and ”simulannealbnd”. Firstly, starting from an initial point $\mathbf{p}_0$, ”simulannealbnd” operated by Simulated Annealing algorithm [54] was used as an global optimizer to get rough initial point $\mathbf{p}_r$ as shown in Figure 4.10. After that, we used ”fmincon” operated by Interior point algorithm [55] to get final solution $\hat{\mathbf{p}}$ more precisely starting from the rough initial point $\mathbf{p}_r$.

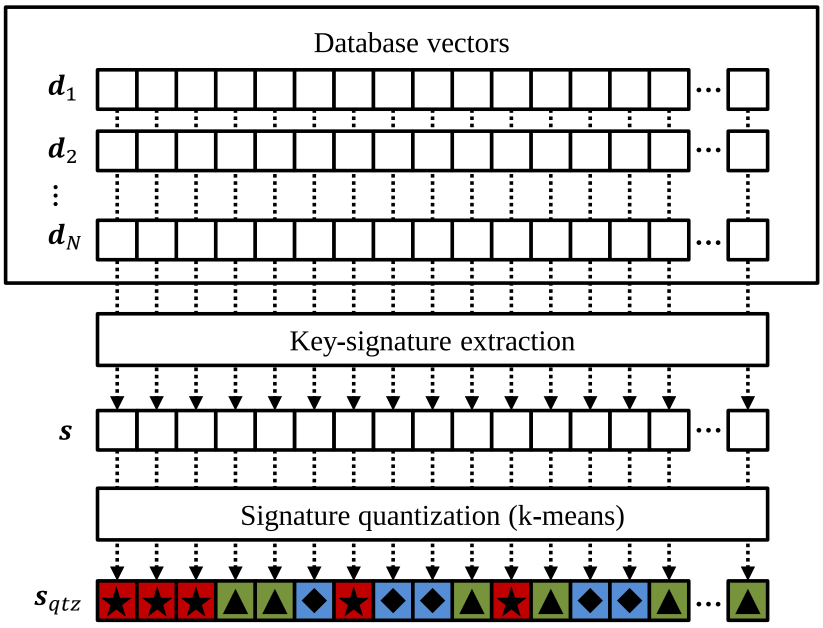


Figure 4.8: Illustrations of key-signature extraction and key-signature quantization. $\mathbf{d}_n (\in \Re^K)$ means a BoW vector of image n with codebook size K. Each square of a BoW vector means each dimension, in other words, that means each codeword or BoW histogram bin. $\mathbf{s} (\in \Re^K)$ is the key-signature vector containing key-signature values for each codeword. After conducting 1-dimensional k-means clustering with $k = \alpha$ using elements of $\mathbf{s}$, we get a dimensionality reduced key-signature vector $\mathbf{s}_{qtz} (\in \Re^\alpha)$ containing α quantized key-signatures.

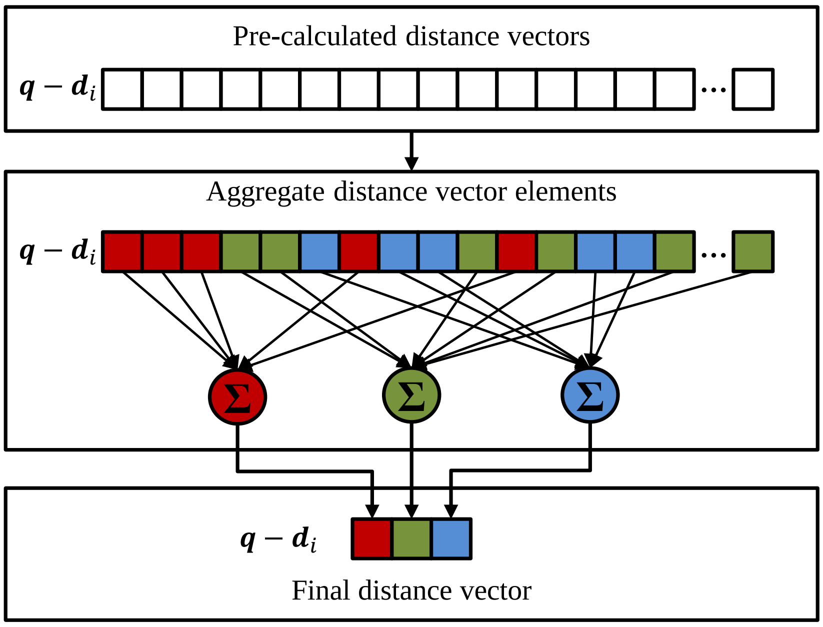


Figure 4.9: Illustrations of generating dimensionality reduced distance vector. A square denotes a codeword and an array of squares denotes a distance vector between two BoW histograms $\mathbf{q}$ and $\mathbf{d}_i$. Squares with each color means that they are members of a same cluster generated by key-signature clustering in Figure 4.8. The circles with $\sum$ denote operations of summation.

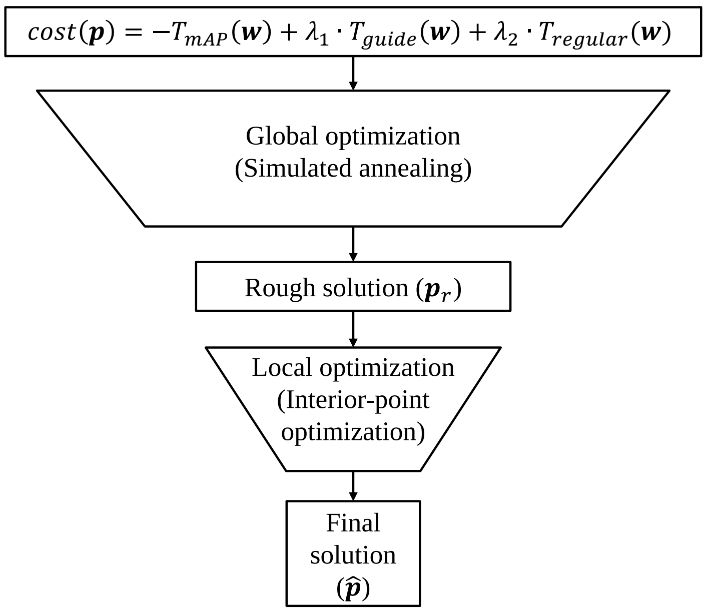

$cost(\boldsymbol{p}) = -T_{mAP}(\boldsymbol{w}) + \lambda_1 \cdot T_{guide}(\boldsymbol{w}) + \lambda_2 \cdot T_{regular}(\boldsymbol{w})$
Global optimization (Simulated annealing)
Rough solution ($\boldsymbol{p}_r$)
Local optimization (Interior-point optimization)
Final solution ($\hat{\boldsymbol{p}}$)

Figure 4.10: Optimization strategy. $\mathbf{p}$ denotes shape parameters (Table 4.1) determining the shape of WMF and $\mathbf{p}_r$ denotes rough solution obtained by simulated annealing [54]. $\hat{\mathbf{p}}$ is the final solution obtained by Interior-point algorithm [55] from the starting point $\mathbf{p}_r$.

4.5 Codewords Weighting

We trained WMF for each signature type by using the method in Subsection 4.4 with respect to three types of key-signature, document frequency, level of hierarchical codebook and SNSTD. We train WMFs with the same method regardless of key-signature type, however, method for obtain weight vectors from WMF and applying the weight vector is different depending on whether a weight vector is intra-class or not. As shown in Table 4.2, a key-signature vector of document frequency or level will be converted to a single global weight vector, on the other way, a key-signature of SNSTD of a class will be converted to a intra-class weight vector for the class. For this reason, different strategies for weighting method is needed according to global or intra-class codewords weighting.

Global Codewords Weighting If we define C key-signatures for global codewords weighting, we can convert the C key-signature vectors to C global weight vectors for a codebook. We apply a set of C global weight vectors $W = \{\mathbf{w}_c | 1 \leq c \leq C\}$ to a pre-normalized query vector $\mathbf{q} = \begin{bmatrix} q_1, & q_2, & \cdots, & q_K \end{bmatrix}^\top$ (BoW histogram) and a database vector $\mathbf{d} = \begin{bmatrix} d_1, & d_2, & \cdots, & d_K \end{bmatrix}^\top$ (BoW histogram):

$$\begin{aligned} q_k &:= q_k \cdot \textstyle\prod_{c=1}^{C} w_k^c, \\ d_k &:= d_k \cdot \textstyle\prod_{c=1}^{C} w_k^c, \end{aligned} \tag{4.14}$$

and then we get a weighted query $\mathbf{q}_{wtd}$ and a weighted database vector $\mathbf{d}_{wtd}$. After that, we conduct scoring of the database vector like [6]:

$$score(\mathbf{q}, \mathbf{d}) = \|\mathbf{q} - \mathbf{d}\|_1 , \tag{4.15}$$

where $score(\mathbf{q}, \mathbf{d})$ is the scoring function of $\mathbf{d}$ and $\mathbf{q}$ of which the value becomes smaller as the similarity gets larger because $\|.\|_1$ returns L_1-norm. Note that we do not normalize the weighted vectors because we train the WMF without normalization after applying a weight vector.

Intra-class Codewords Weighting The point of intra-class codewords weighting was that the key words of images differ from class to class. Even if we have made optimal weight vectors through indexing codewords using SNSTDs for each class, however, we encounter a significant problem that we do not know the class of the query image. In other words, in query time, we do not know which weight vector we have to apply because we do not the class of the query. We propose a solution that, we assume the database is pre-categorized, we make subsets from the query without intra-class codewords weighting and re-ranking them as shown in Figure 4.11. In the case of intra-class codewords weighting, since we assume that database images are pre-categorized, we can make a set of candidate classes for the query and obtain a set of intra-class weight vectors according to the candidate classes. We then apply the intra class weight vectors into a set of every images corresponding to the candidate classes. After that, we re-rank the weight-applied images and get final results.

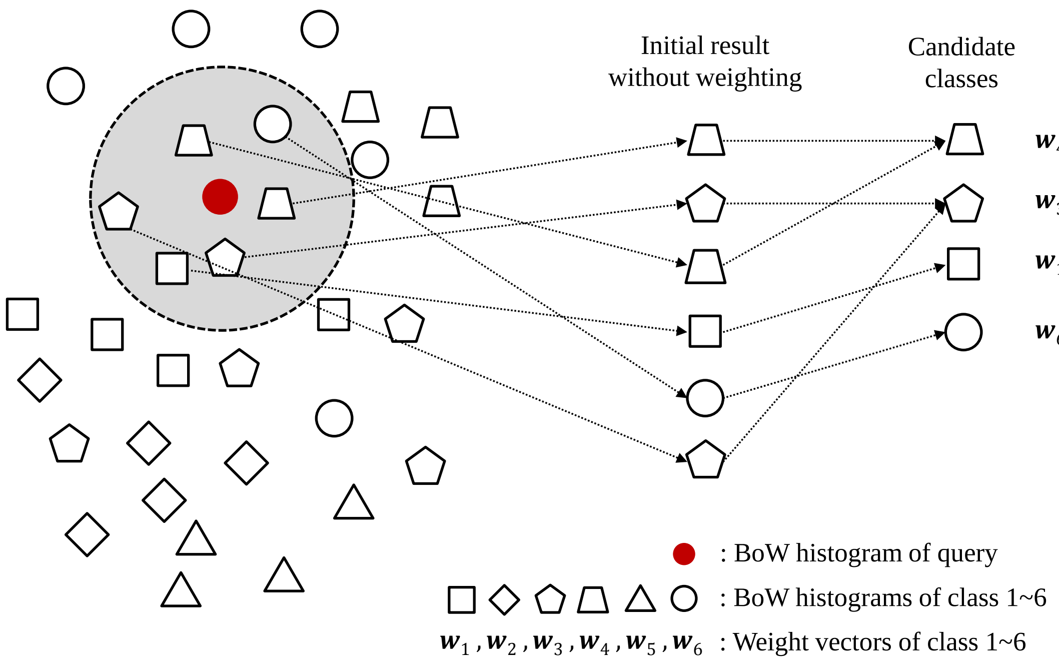

Initial result without weighting
Candidate classes
w_4
w_3
w_1
w_6
● : BoW histogram of query
: BoW histograms of class 1~6
$w_1, w_2, w_3, w_4, w_5, w_6$: Weight vectors of class 1~6

Figure 4.11: Illustrations of BoW histogram space, in which it shows procedures of obtaining sub-classes and selecting intra-class weight vectors to be used for re-ranking. Initial result is obtained by k-nearest neighborhoods without any intra-class codewords-weighting scheme. Each data has its class information (label) and a corresponding weight vector because of the assumption that database images are pre-categorized.

Chapter 5. Experimental Result

To evaluate the performance of proposed method, we conducted various experiments with two dataset. We will first introduce the dataset we used in Section 1, experiments with document frequency as a key-signature of WMF in Section 2 and experiments using level in Section 3. Last, we will introduce the result of applying document frequency and level to the weight in Section 4. We show various experimental result with tables and figures in which some symbols are used to denote various weighting method. The descriptions of symbols to be used are in Table 5.1.

Method symbol	Description
None	No weight vector applied (uniform weight) and L_1-normalized.
s(org)	Weighted by original WMF of key-signature type s and L_1-normalized.
(For example, df(org))	(Weighted by original tf-idf weighting and L_1-normalized. [2])
(For example, level(org))	(Not weighted according to levels and L_1-normalized. [6])
s(Exp)	Weighted by $WMF_{Exp}(.)$ of key-signature type s.
s(Poly2)	Weighted by $WMF_{Poly}(.)$ with order 2 of key-signature type s.
s(LUT)	Weighted by $WMF_{LUT}(.)$ of key-signature type s.
$\mathrm{s}_1(\mathrm{WMF}_A)+\mathrm{s}_2(\mathrm{WMF}_B)$	$\mathrm{s}_2(\mathrm{WMF}_B)$ is applied after applying the method $\mathrm{s}_1(\mathrm{WMF}_A)$.

Table 5.1: Description of symbols which are used in experiential result.

5.1 Datasets and Experimental Setup

Under the assumption that different WMF will be learnt depending on the property of the database we want to retrieve, we experiments with two datasets with different properties, UKbench [6] and Oxford5K [8].

UKbench dataset UKBenceh dataset [6] is composed of 2550 classes of objects or scene and total 10200 images. In other words, each class is composed of 4 images. Four images of each class that were near-duplicated with a slight change in view point toward the object or scene as shown in the left side of Figure 5.1. The retrieval performance is measured by mAP and Top-4. Top-4 means average number of ground truth image among the first four returned images. We sampled 300 classes and trained WMF as default, then conducted test on remaining 2250 (=2250−300) classes.

Oxford5K dataset Oxford5K dataset [8] has a very diverse view points and illumination changes on the Oxford buildings taken outside as shown in the right side of Figure 5.1. The images are regarded as ground truth only if they are more than 25% overlapped with the query buildings. It is composed of 5062 images in total, with 55 queries consisting 5 images per landmark (label) for 11 different landmarks. The retrieval performance is measured by mAP. Since Oxford5K has only 11 classes, we sampled one query per class and trained WMF for 5062 images as default. We then conducted test on remaining 5051 (=5062−11) images.

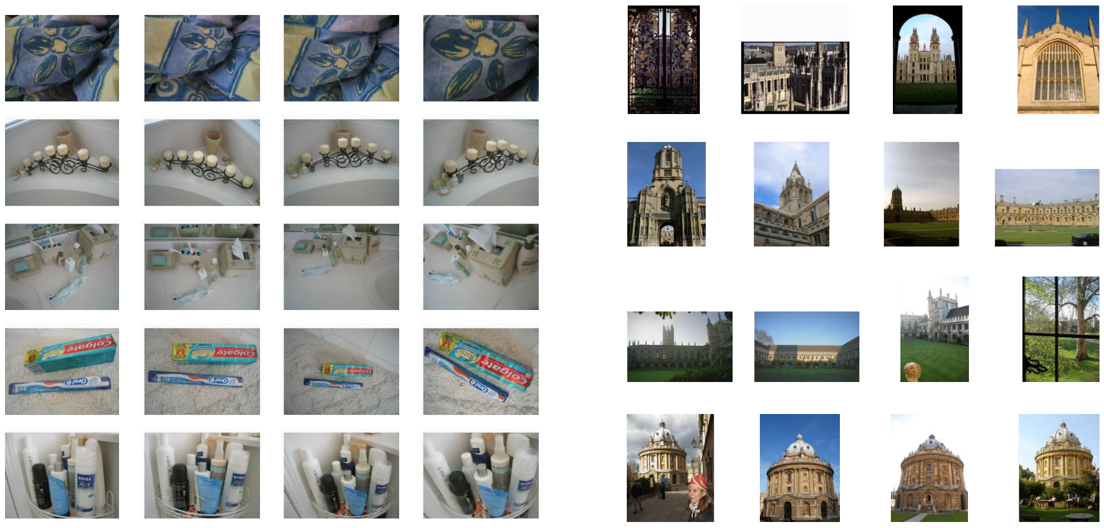


Figure 5.1: Sample images of UKbench dataset (left) and Oxford5K dataset (right). Images of each row belong to a same class.

5.2 Key-signature: Document Frequency

In this section, we shows the experimental result of learning WMF with document frequencies. We experimented with both flat codebook and hierarchical codebook.

Experiment on Flat Codebook We utilized on-line-available data to evaluate WMF with df as a key-signature in flat codebook. We downloaded and conducted an experiment with a 100k-size codebook from Flickr60K. The codebook was trained with the features which detected the interesting regions with Hessian-Affine extractor and described them with SIFT [1]. We used the same feature as the one we used to produce a codebook. For Oxford5K, we downloaded the BoW histogram using a codebook size of 1M which is provided in this dataset website, and used it without any transformation. Table 5.2 and Table 5.3 shows the result of the experiment with flat codebook on UKbench and Oxford5Kd dataset respectively. When we used $WMF_{poly2}(\cdot)$ trained with the intact data, tf-idf weighting slightly outperformed. When we used $WMF_{poly2}(\cdot)$ trained with data which tf-idf weighting was applied to, however, mAP showed improvement by 2.05% compared to no weighting and by 1.14% compared to tf-idf weighting. This is almost twice of the improvement of tf-idf weighting compared to no weighting, which is 0.91%. Oxford5K dataset showed similar result. While tf-idf weighting slightly outperformed when we used $WMF_{poly2}(\cdot)$ trained with the intact data, mAP showed improvement by 1.48% compared to no weighting and by 0.79% compared to tf-idf weighting when we used $WMF_{poly2}(\cdot)$ trained with data which tf-idf weighting was applied to.

We found a very interesting result from Figure 5.3, 5.4 where trained WMFs are plotted. The result contradicts with the conventional concept of tf-idf weighting. The tendency is especially clear in the result of training $WMF_{Exp}(\cdot)$. Nevertheless, the $WMF_{Exp}(\cdot)$s outperformed the conventional tf-idf weighting, like in Table 5.2 and Tabel 5.3. With this experiment, we verified that the assumption in text domain does not necessarily accord

Dataset	UKbench					
Key-signature	Document frequency					
Weighting method	None	df(org)	**df(Poly2)**	**df(LUT)**	df(org)+ **df(Poly2)**	df(org)+ **df(LUT)**
mAP(%)	82.5	83.41	83.88	83.87	**84.55**	84.28
Top-4	2.92	2.97	2.99	2.99	**3.03**	3.01

Table 5.2: Retrieval performance on the UKbench using various weighting strategy. The key-signature type is document frequency.

Dataset	Oxford5K					
Key-signature	Document frequency					
Weighting method	None	df(org)	**df(Poly2)**	**df(LUT)**	df(org)+ **df(Poly2)**	df(org)+ **df(LUT)**
mAP(%)	61.64	61.84	62.36	62.58	62.63	**62.79**

Table 5.3: Retrieval performance on the Oxford5K using various weighting strategy. The key-signature type is document frequency.

with image domain. Rather, assigning larger weight to words with larger df improves performance. By the same means of Figure 2.7, Figure 5.2 visualizes only matched line with codewords having relatively large document frequencies. As shown in the example, in image domain, codewords having large document frequencies can help image matching in contrast with text domain. Moreover, the result accords with our prediction that the shape of WMF will differ depending on the property of given database. Especially in the result of experiment training $WMF_{LUT}(\cdot)$, the largest weight is assigned to different dfs in UKbench and Oxford5K.

Table 5.4 showes retrieval performance depending on the number of training classes in UKbench dataset. Although the rate of training class sampling changes, improvement with respect to no weighting does not change a lot. Note that even though we learn a WMF only with a small set of training data, sampling rate lower then 1%, we can get clear improvement compared to original tf-idf weighting scheme.

Experiment on Hierarchical Codebook We conducted evaluation in each level of hierarchical codebook which used df as a key-signature. We used SIFT [1] as a local detector and descriptor and trained hierarchical codebook for each dataset. Here, we set the number of branch 10 and the depth 6. Table 5.5, 5.6 and Figure 5.5, 5.6 show the retrieval performance in hierarchical codebook. WMF training after tf-idf weighting outperformed tf-idf weighting in every level used in hierarchical codebook.

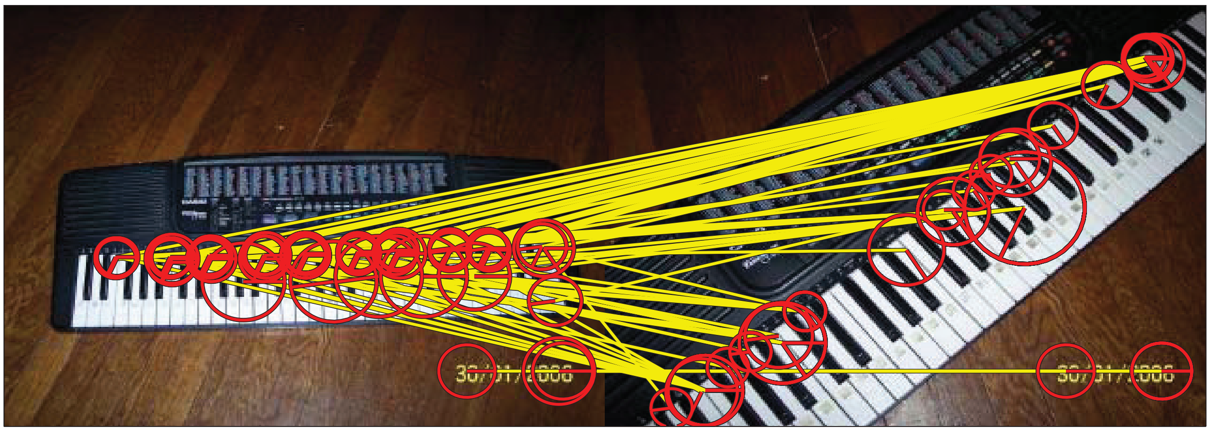


Figure 5.2: Visualizations of matched codewords with high document frequencies in UKbench dataset. The red blobs denote detectetd regions with a SIFT detector [1]. A yellow line denotes that the two detected regions are matched to the same codewords.

Dataset		UKbench									
Key-signature		Document frequency									
Weighting method		None		df(org)				df(org)+**df(LUT)**			
# training classes	Sampling rate	mAP (%)	Top-4	mAP (%)	ΔmAP (%)	Top-4	ΔTop-4	mAP (%)	ΔmAP (%)	Top-4	ΔTop-4
100	0.98	82.15	2.9	83.05	+ 0.9	2.96	+ 0.06	84.18	+ **2.03**	3.01	+ **0.11**
300	2.94	82.5	2.92	83.41	+ 0.91	2.97	+ 0.05	84.28	+ **1.78**	3.01	+ **0.09**
500	4.90	82.65	2.93	83.61	+ 0.96	2.98	+ 0.05	84.8	+ **2.15**	3.04	+ **0.11**

Table 5.4: Retrieval performance on UKbench according to various sampling strategy. Δ means the performance improvement with respect to no weighting. The key-signature type is df.

Dataset	UKbench											
Key-signature	Document frequency											
Method	None		df(org)		**df(Poly2)**		**df(LUT)**		df(org)+**df(Poly2)**		df(org)+**df(LUT)**	
level	mAP (%)	Top-4	mAP (%)	Top-4	mAP (%)	Top-4	mAP (%)	Top-4	mAP (%)	Top-4 (%)	mAP	Top-4
4	76.79	2.61	77.71	2.65	78.65	2.69	78.55	2.68	**79.8**	**2.75**	79.73	2.74
5	79.73	2.76	80.652	2.8	81.84	2.86	81.65	2.85	**82.46**	**2.89**	82.37	2.88
6	82.85	2.92	83.48	2.96	83.84	2.97	83.72	2.96	**84.34**	**3.0**	84.16	2.99

Table 5.5: Retrieval performance on UKbench dataset according to used level of hierarchical codebook. The key-signature type is df.

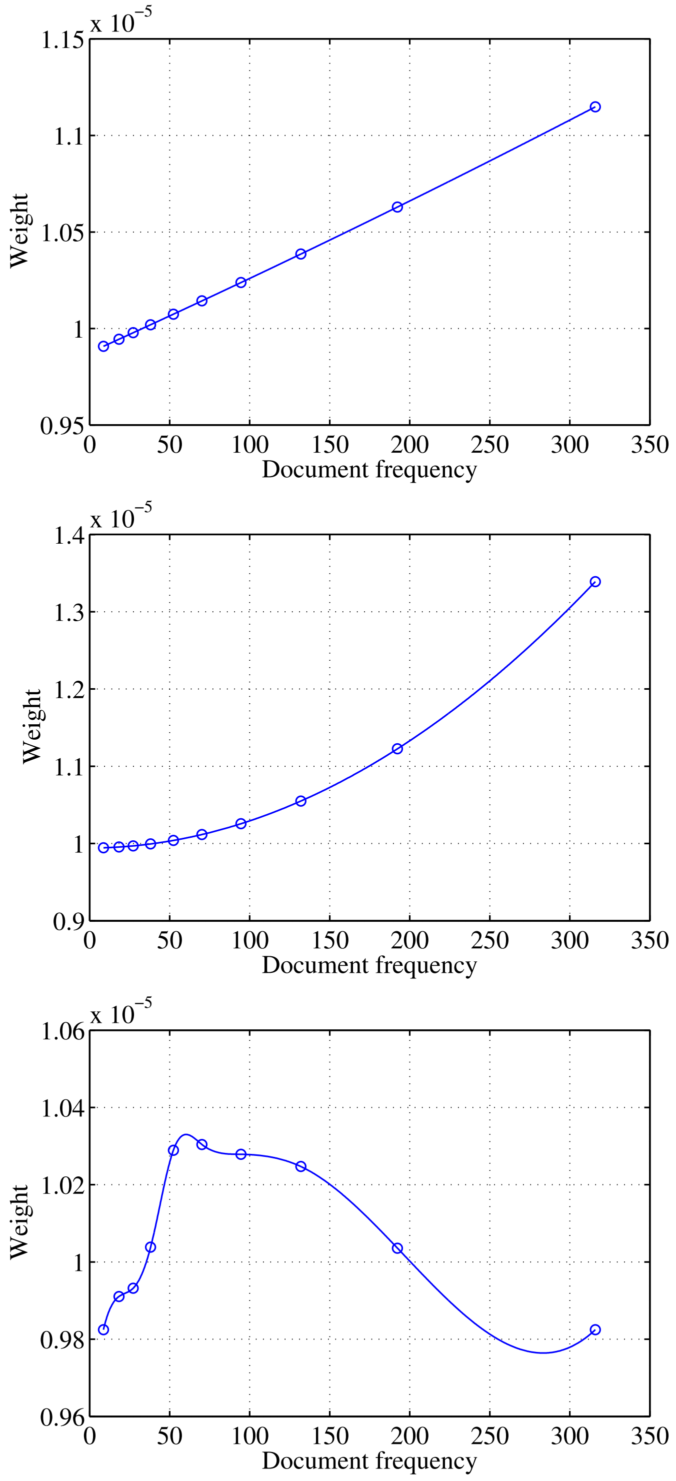


Figure 5.3: WMFs trained on the UKbench dataset. From top to bottom, each figure is about $WMF_{Exp}(\cdot)$, $WMF_{poly2}(\cdot)$ and $WMF_{LUT}(\cdot)$ respectively. $WMF_{LUT}(\cdot)$ is interpolated by blue line for visualization. The key-signature type is df.

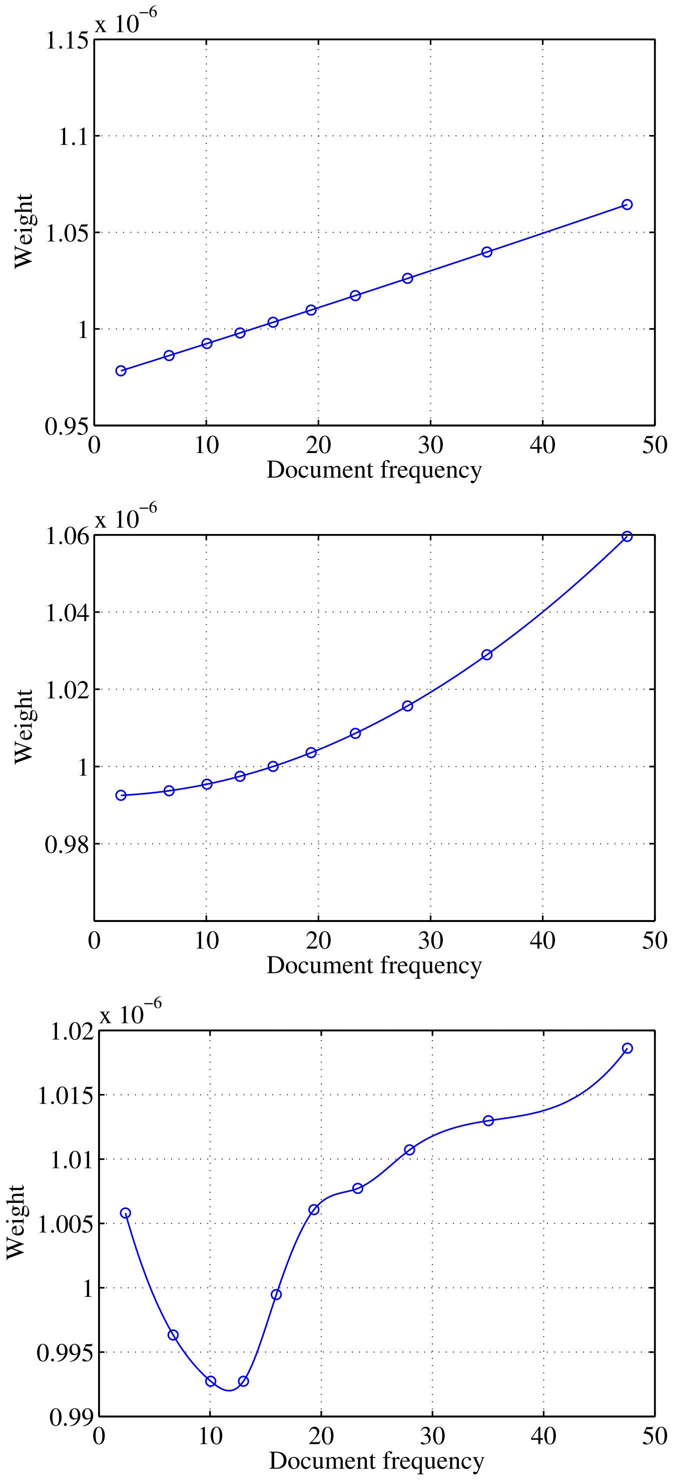


Figure 5.4: WMFs trained on the Oxford5K dataset. From top to bottom, each figure is about $WMF_{Exp}(\cdot)$, $WMF_{poly2}(\cdot)$ and $WMF_{LUT}(\cdot)$ respectively. $WMF_{LUT}(\cdot)$ is interpolated by blue line for visualization. The key-signature type is df.

Dataset	Oxford5K					
Key-signature	Document frequency					
Method	None	df(org)	**df(Poly2)**	**df(LUT)**	df(org)+**df(Poly2)**	df(org)+**df(LUT)**
Level	mAP(%)	mAP(%)	mAP(%)	mAP(%)	mAP(%)	mAP(%)
4	35.61	36.8	36.81	36.6	37.78	**38.12**
5	45.42	46.78	46.59	46.0	47.74	**47.84**
6	50.63	51.29	50.67	51.21	51.62	**51.91**

Table 5.6: Retrieval performance on Oxford5K dataset according to used level of hierarchical codebook. The key-signature type is *df*.

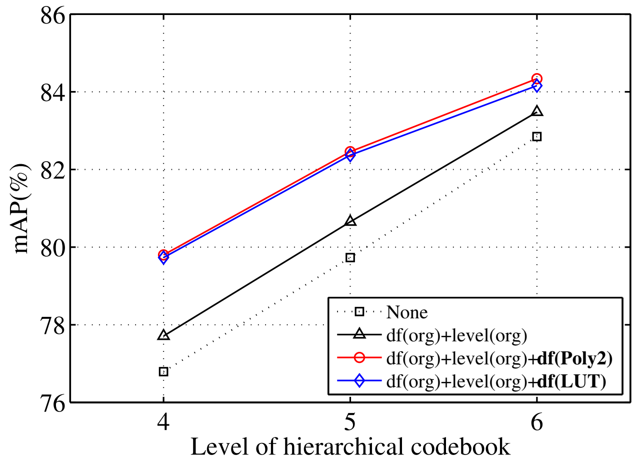


Figure 5.5: Retrieval performance on UKbench dataset using various weighting strategy according to used level of hierarchical codebook. The key-signature type is *df*.

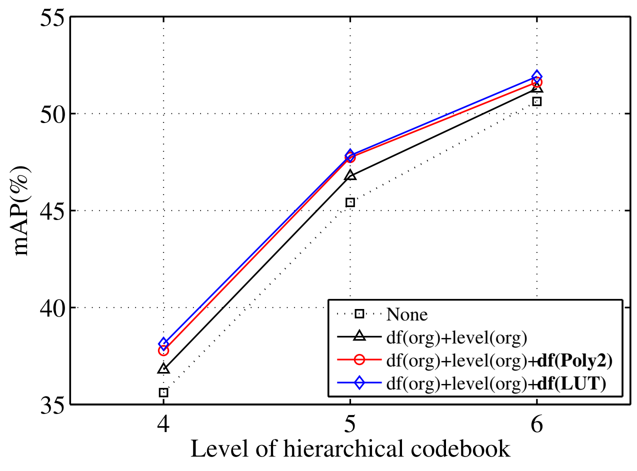


Figure 5.6: Retrieval performance on Oxford5K dataset using various weighting strategy according to used level of hierarchical codebook. The key-signature type is *df*.

5.3 Key-signature: Level of Hierarchical Codebook

We trained WMF with level of hierarchical codebook as a key-signature and conducted experimental setup of hierarchical codebook same with Section 5.2. The result in Table 5.8 and Figure 5.7, 5.8 shows that our method outperforms conventional hierarchical scoring [6] which only applies tf-idf weighting for each level. In case of UKbench, the performance increases as the number of levels considered increases but decreases if only two levels are considered. Especially, in case of Oxford dataset, the performance significantly decreases as the number of levels considered increases. However, our method shows generally good performance as the number of levels considered increases.

Figure 5.9, 5.10 shows WMF obtained with level as the key-signature. The larger the level is, that is, the closer the level is to the leaf, the discriminability gets better and as a result the two datasets are both trained as an increasing function. Note that, however, not too small levels do not have zero weights because it takes a role as compensating the problems of sensitivity to noisy descriptors in large levels.

As shown in Table 5.9, WMF with level as the key-signature also shows that improvement with respect to [6] does not change very much although the sampling rate of training class changes. Note that even though we learned a WMF only with a small set of training data, sampling rate lower then 1%, we got most significant improvement compared to the results of higher sampling rates.

Dataset	UKbench					
Key-signature	Level					
Weighting method	df(org)		df(org)+**level(Exp)**		df(org)+**level(LUT)**	
levels considered	mAP(%)	Top-4	mAP(%)	Top-4	mAP(%)	Top-4
5, 6	83.08	2.93	83.98	2.97	**84.0**	**2.98**
4, 5, 6	83.98	2.95	84.84	3.02	**85.13**	**3.04**
3, 4, 5, 6	85.66	3.06	86.45	3.11	**87.02**	**3.14**
2, 3, 4, 5, 6	86.12	3.08	87.67	3.17	**88.73**	**3.22**
1, 2, 3, 4, 5, 6	85.72	3.07	87.95	3.19	**88.73**	**3.22**

Table 5.7: Retrieval performance on UKbench using various number of levels from the leaf for hierarchical scoring. The key-signature type is level.

Dataset	Oxford5K		
Key-signature	Level		
Weighting method	df(org)	df(org)+**level(Exp)**	df(org)+**level(LUT)**
levels considered	mAP(%)	mAP(%)	mAP(%)
5, 6	50.98	51.84	**52.03**
4, 5, 6	49.11	**52.13**	50.97
3, 4, 5, 6	49.4	52.05	**52.38**
2, 3, 4, 5, 6	49.4	**52.25**	49.36
1, 2, 3, 4, 5, 6	49.04	**52.25**	49.1

Table 5.8: Retrieval performance on Oxford5K using various number of levels from the leaf for hierarchical scoring. The key-signature type is level.

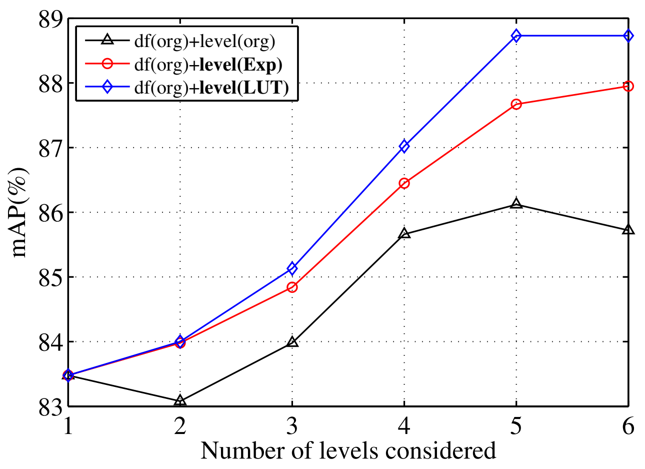


Figure 5.7: Retrieval performance on UKbench dataset using various number of levels from the leaf for hierarchical scoring. The key-signature type is level.

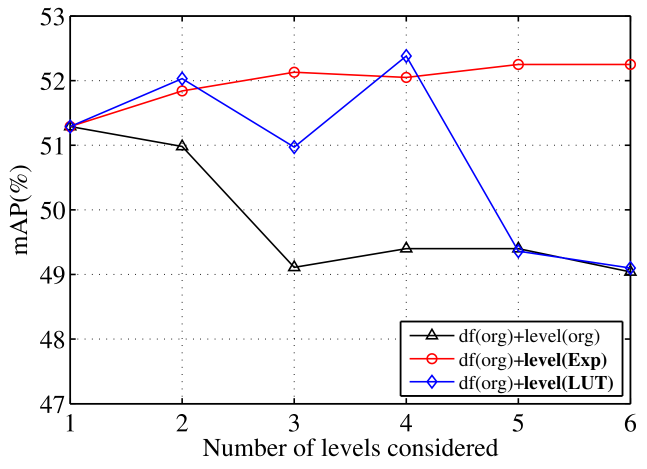
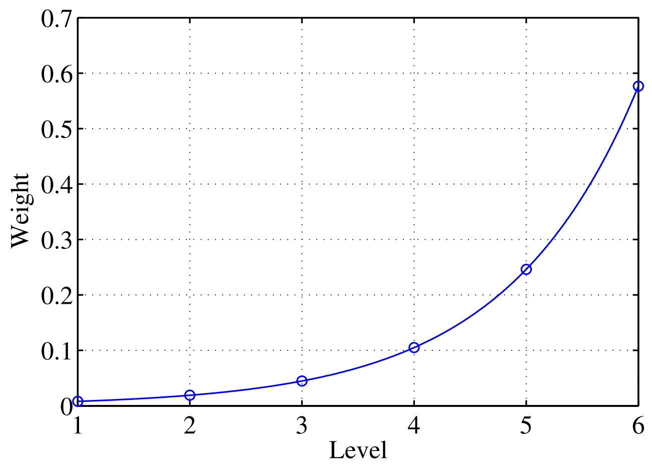
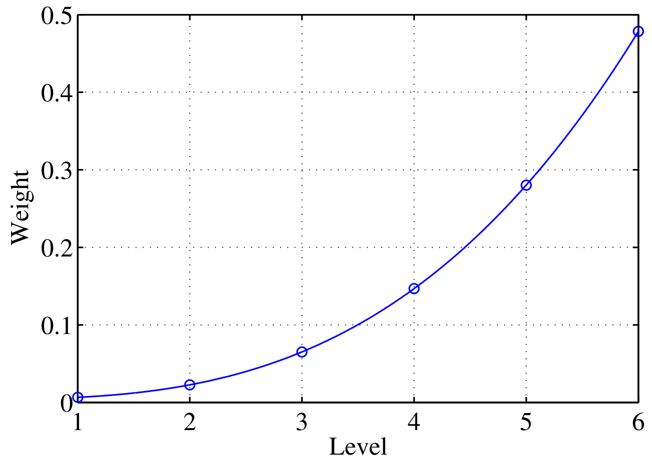


Figure 5.8: Retrieval performance on Oxford5K dataset using various number of levels from the leaf for hierarchical scoring. The key-signature type is level.

Dataset		UKbench					
Key-signature		Level					
Weighting method		df(org)+level(org)		df(org)+**level(LUT)**			
# training classes	Sampling rate (%)	mAP(%)	Top-4	mAP(%)	ΔmAP(%)	Top-4	ΔTop-4
100	0.98	85.65	3.06	87.84	**2.19**	3.17	**0.11**
300	2.9412	85.66	3.06	87.02	**1.36**	3.14	**0.08**
500	4.9020	86.16	3.09	88.2	**2.051**	3.18	**0.09**

Table 5.9: Retrieval performance on UKbench dataset according to various sampling strategy. Δ means the performance improvement. The key-signature type is level.





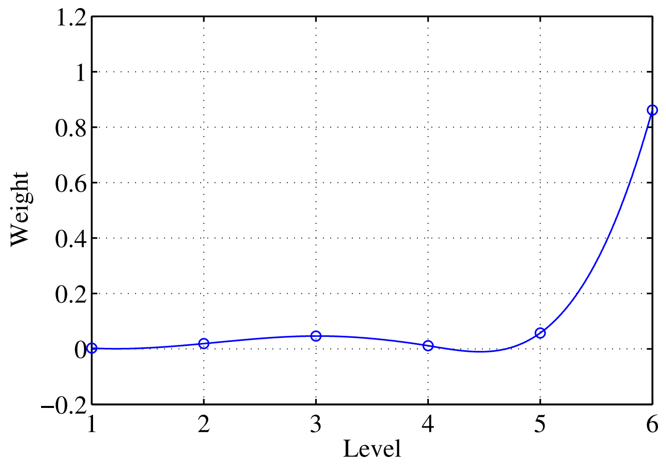


Figure 5.9: WMFs trained on the UKbench dataset. From top to bottom, each figure is about $WMF_{Exp}(\cdot)$, $WMF_{poly2}(\cdot)$ and $WMF_{LUT}(\cdot)$ respectively. $WMF_{LUT}(\cdot)$ is interpolated by blue line for visualization. The key-signature type is level.

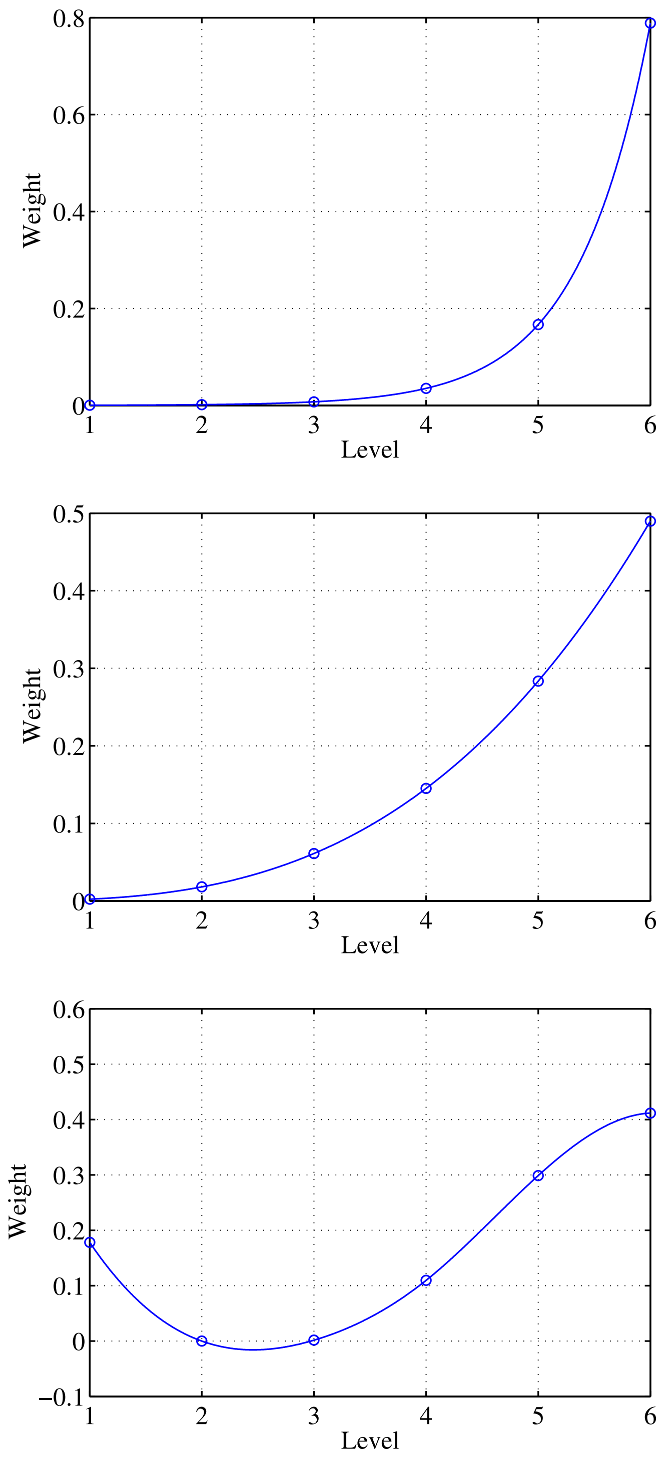


Figure 5.10: WMFs trained on the Oxford5K dataset. From top to bottom, each figure is about $WMF_{Exp}(\cdot)$, $WMF_{poly2}(\cdot)$ and $WMF_{LUT}(\cdot)$ respectively. $WMF_{LUT}(\cdot)$ is interpolated by blue line for visualization. The key-signature type is level.

5.4 Key-signature: Both Document Frequency and Level

For the case with both df and level as the key-signature, we set our experiment up like the experimental setup of hierarchical codebook in Section 5.2. With df as the key-signature, we trained WMF_{LUT} in each level of the hierarchical codebook and applied weight to the training vectors according to (4.14). Considering the weighted data as source, we trained WMF_{Exp} with level as the key-signature and obtained the final result. In Table 5.10, 5.11 and Figure 5.11, 5.12, we can see that there was bigger improvement generally in the performance than the improvement only with our document frequency weighting (Section 5.2). Especially, in the results of UKbench dataset (Table 5.10 and Figure 5.11), the performances were improved with repect to the reults only with our document frequency weighting on every number of levels considered.

Dataset	UKbench			
Key-signature	Document frequency and level			
Weighting	df(org)+level(org)		df(org)+**df(LUT)+level(Exp)**	
Levels	mAP	Top-4	mAP	Top-4
5, 6	83.08	2.93	**85.4**	**3.04**
4, 5, 6	83.98	2.95	**86.29**	**3.09**
3, 4, 5, 6	85.66	3.06	**87.68**	**3.16**
2, 3, 4, 5, 6	86.12	3.08	**88.5**	**3.21**
1, 2, 3, 4, 5, 6	85.72	3.07	**88.61**	**3.21**

Table 5.10: Retrieval performance on UKbench and Oxford5K using various number of levels from the leaf for hierarchical scoring. The key-signature type is document frequency and level. WMF_{Exp} is for level weighting and WMF_{LUT} is for document frequency weighing.

Dataset	Oxford5K	
Key-signature	Document frequency and level	
Weighting	df(org)+level(org)	df(org)+**df(LUT)+level(Exp)**
Levels	mAP(%)	mAP(%)
5, 6	50.98	**52.4**
4, 5, 6	49.11	**52.45**
3, 4, 5, 6	49.4	**51.74**
2, 3, 4, 5, 6	49.4	**52.04**
1, 2, 3, 4, 5, 6	49.04	**51.55**

Table 5.11: Retrieval performance on UKbench and Oxford5K using various number of levels from the leaf for hierarchical scoring. The key-signature type is document frequency and level. WMF_{Exp} is for level weighting and WMF_{LUT} is for document frequency weighing.

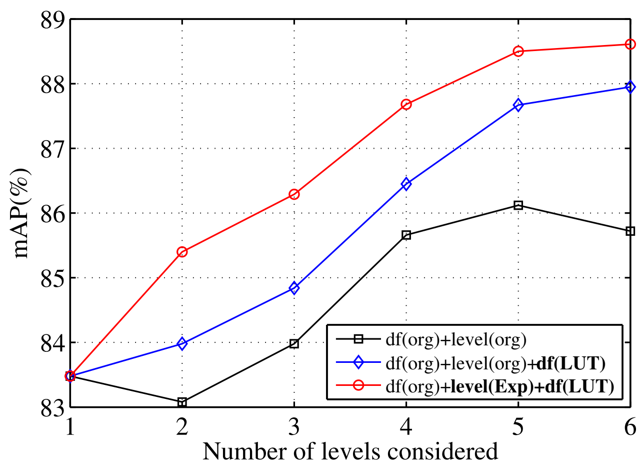


Figure 5.11: Retrieval performance on UKbench dataset using various number of levels from the leaf for hierarchical scoring. The key-signature type is document frequency and level. WMF_{Exp} is for level weighting and WMF_{LUT} is for document frequency weighing.

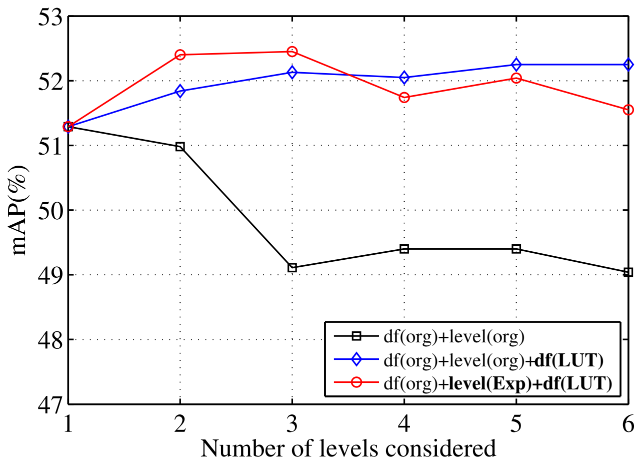


Figure 5.12: Retrieval performance on Oxford5K dataset using various number of levels from the leaf for hierarchical scoring. The key-signature type is document frequency and level. WMF_{Exp} is for level weighting and WMF_{LUT} is for document frequency weighing.

5.5 Key-signature: Intra-Class SNSTD

We conducted an experiment to evaluate intra-class codewords weighting method with intra-class SNSTD as a key-signature type. We experimented it only with UKbench dataset because Oxford5K dataset has too little number of classes (11), therefore, the procedure of obtaining candidate classes does not meaningful with the dataset. We used a hierarchical codebook [6] with depth of 6 and branch factor of 10. The intra-class codewords weighting method was applied after the conventional tf-idf weighting method [6, 2]. We randomly sampled 200 classes among the total 2550 classes to utilize it as training data to learn WMF. We used exponential function type in Table 4.1 to represent WMF (WMF_{Exp}). The retrieval performance is measured by Top-x which is average number of ground truth images among top-x images retrieved.

Figure 5.13 shows the retrieval performance. As shown from the result, intra-class codewords weighting outperforms [6] regarding every x. The result of the experiments conducted with various pre-defined number of candidate classes is shown in Figure 5.14. In here, likewise, the method outperforms [6] regarding every number of candidate classes as well. However, with the bigger the number of candidate classes, the performance score gets lower in principle because the number of subset images to be re-ranked becomes larger.

Figure 5.15 shows WMF obtained with intra-class SNSTD as the key-signature. The smaller the intra-class SNSTD is, the discriminability gets better and as a result WMF on UKbench dataset trained as an decreasing function.

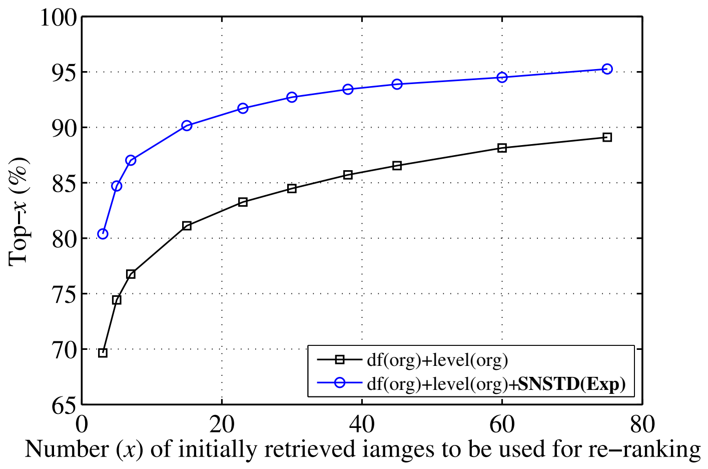


Figure 5.13: Retrieval performances on UKbench dataset with various number x of top retrieved images. The performances are measured by Top-x denoting average number of ground truth images among top x images. The key-signature type is intra-class SNSTD.

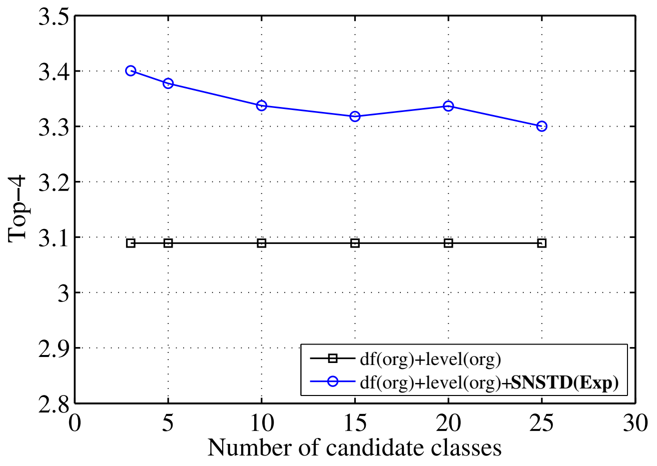


Figure 5.14: Retrieval performances on UKbench dataset with various pre-defined number of candidate classes. The performances are measured by Top-4. The key-signature type is intra-class SNSTD.

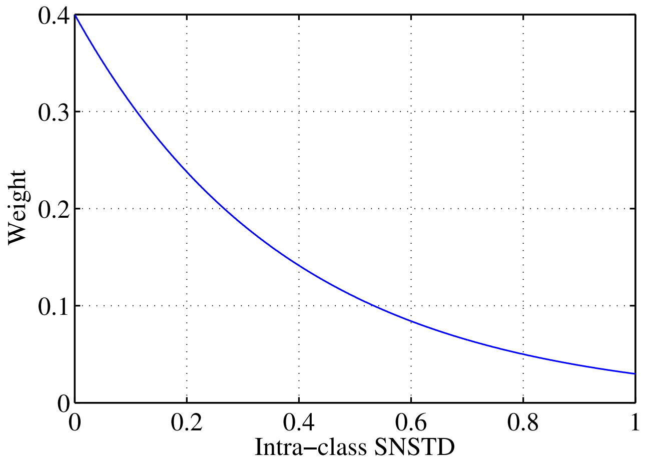


Figure 5.15: A WMF ($WMF_{Exp}(\cdot)$) trained on UKbench dataset. The key-signature type is intra-class SNSTD.

Chapter 6. Conclusion

tf-idf weighting and hierarchical scoring, which is a commonly used codebook weighting scheme and scoring method in BoW representation based image retrieval, is usually understood to improve retrieval performance with a simple procedure. However, the degree of improvement is not too significant. For some dataset, rather, it sometimes brings a problem of worsening precision. Not only that, because recent image retrieval system uses very high dimensional BoW representation, it is regarded impossible to use the fact that different classes have different key words. Only a weight or a same standard is applied to every image classes.

In this thesis, we trained a relevant global weight vector for a codebook only using the information of document frequencies or levels of codewords without any additional cue and obtained an increased discriminability by applying relevant weight to each codeword under the assumption of BoWs representation that codewords occur independently in one image. Moreover, we trained intra-class weight vectors for each class only using the information of statistical codewords appearance within each class under the same assumption. Although the dimensionality of an image described with a very large codebook is very high, we produced a global weight vector or intra-class weight vectors through few minutes of training and there was nearly no additional cost in applying the weight vector in on-line phase. With the proposed method using document frequency or level, the performance is improved compared to existing tf-idf weighting and hierarchical scoring without any additional information used. With the proposed method of weighting intra-class key words, the result shows noticeable improvement compared to the retrieval without intra-class weighting. In the learned WMF of document frequency, especially, we discovered an interesting fact that codewords with high document frequencies are regarded important and it actually improves retrieval performance with large weight.

The reason why the proposed weighting scheme is meaningful is that there remains potential to increase performance if we find another type of key-signature besides document frequency and level of hierarchical codebook. Additional studies to find and apply other key-signatures besides the two types of key-signatures used in the proposed method is needed. In addition to that, it is desirable to train $(n + 1)$-dimensional WMF for n number of types of key-signature, rather than independently training n number of 2-dimensional WMFs for n types of key signature. Furthermore, methodological analysis and thoughtful discussions are needed to understand the reason why codewords with high document frequency are regarded important in image domain.

References

[1] David G. Lowe, Distinctive image features from scale-invariant keypoints, In *International Journal of Computer Vision (IJCV)*, 2004.

[2] J. Sivic and A. Zisserman, Video Google: A Text Retrieval Approach to Object Matching in Videos, In *Proceedings of International Conference on Computer Vision (ICCV)*, 2003.

[3] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, Lost in quantization: Improving particular object retrieval in large scale image databases, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[4] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences, *Journal of Machine Learning Research (JMLR)*, Vol.4, pp.933-969, 2003.

[5] Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang, Spatial-Bag-of-Features, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[6] D. Nister and H. Stewenius, Scalable recognition with a vocabulary tree, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[7] Tighe, Joseph and Lazebnik, Svetlana., Superparsing: scalable nonparametric image parsing with superpixels. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2010

[8] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[9] O. Chum, A. Mikulik, M. Perdoch, J. Matas, Total recall II: Query expansion revisited, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011

[10] X. Wang, M. Yang, T. Cour, S. Zhu, K. Yu, and Tony X. Han, Contextual Weighting for Vocabulary Tree based Image Retrieval, In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011.

[11] H. Cai, F. Yan, and K. Mikolajczyk, Learning Weights for Codebook in Image Classification and Retrieval, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[12] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, Aggregating local image descriptors into compact codes, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2011.

[13] H. Jegou, M. Douze, and C. Schmid, On the burstiness of visual elements, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[14] H. Jegou, M. Douze, and C. Schmid, Improving bag-of-features for large scale image search, In *International Journal of Computer Vision (IJCV)*, 2010.

[15] H. Jegou, M. Douze, and C. Schmid, Hamming Embedding and Weak Geometric consistency for large-scale image search, In *Proceedings of European Conference on Computer Vision (ECCV)*, 2008.

[16] Z. Wu, Q. Ke, M. Isard, and J. Sun, Bundling features for large scale partial-duplicate web image search, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[17] H. Jegou, H. Harzallah, and C. Schmid, A contextual dissimilarity measure for accurate and efficient image search, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[18] T. Tuytelaars, and C. Schmid, Vector Quantizing Feature Space with a Regular Lattice, In *Proceedings of International Conference on Computer Vision (ICCV)*, 2007.

[19] O. Chum, J. Philbin, and A. Zisserman, Near Duplicate Image Detection: min-Hash and tf-idf Weighting, In *Proceedings of British Machine Vision Conference (BMVC)*, 2008.

[20] 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 *Proceedings of International Conference on Computer Vision (ICCV)*, 2007.

[21] Y. Zhang, Z. Jia, and T. Chen, Image Retrieval with Geometry-Preserving Visual Phrases, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[22] 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 *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[23] R. Arandjelovic, and A. Zisserman, Three things everyone should know to improve object retrieval, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[24] R. Tavenard, H. Jegou, and L. Amsaleg, Balancing clusters to reduce response time variability in large scale image search, In *International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2011.

[25] J. He, J. Feng, X. Liu, T. Cheng, T. Lin, H. Chung and S. Chang, Mobile Product Search with Bag of Hash Bits and Boundary Reranking, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[26] X. Shen, Z. Lin, J. Brandt and Y. Wu, Mobile Product Image Search by Automatic Query Object Extraction, In *Proceedings of European Conference on Computer Vision (ECCV)*, 2012.

[27] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, T. L. Berg, Parsing Clothing in Fashion Photographs, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[28] J. Hays and Alexei A. Efros. Scene Completion Using Millions of Photographs, In *Proceedings of ACM SIGGRAPH*, 2007.

[29] S. Agarwal, N. Snavely, I. Simon, Steven M. Seitz and R. Szeliski, Building Rome in a Day, In *Proceedings of International Conference on Computer Vision (ICCV)*, 2009.

[30] S. Agarwal, Y. Furukawa, N. Snavely, B. Curless, Steven M. Seitz and R. Szeliski, Reconstructing Rome, In *IEEE Computer*, 2010

[31] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, Steven M. Seitz and R. Szeliski, Building Rome in a Day, In *Communications of the ACM (CACM)*, 2011.

[32] A. Torralba, R. Fergus and W. T. Freeman, 80 million tiny images: a large dataset for non-parametric object and scene recognition, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol.30, No.11, pp.1958-1970, 2008.

[33] C. Wengert, M. Douze and H. Jegou, Bag-of-colors for improved image search, In *Proceedings of ACM Multimedia (ACMM)*, 2011.

[34] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1582–1596, 2010.

[35] R. Arandjelovic and A. Zisserman, Smooth Object Retrieval using a Bag of Boundaries, In *Proceedings of International Conference on Computer Vision (ICCV)*, 2011.

[36] M. Varma and A. Zisserman, A statistical approach to texture classification from single images, In *International Journal of Computer Vision (IJCV)*, Vol.62, pp.61-81, 2005.

[37] H. Bay, A. Ess, T. Tuytelaars and L. V. Gool, SURF: Speeded Up Robust Features, *Computer Vision and Image Understanding (CVIU)*, vol.110, No.3, pp. 346-359, 2008.

[38] J. Matas, O. Chum, M. Urban and T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions, In *Proceedings of British Machine Vision Conference (BMVC)*, volume 1, pages 384–393, 2002.

[39] K. Mikolajczyk and C. Schmid, Scale and affine invariant interest point detectors, In *International Journal of Computer Vision (IJCV)*, Vol.60, No.1, pp.63-86, 2004.

[40] G. Salton and C. Buckley, Term-weighting approaches in automatic text retrieval. In *Information Processing and Management (IPM)*, 24(5):513–523, 1988.

[41] H. Jegou, H. Harzallah and C. Schmid, A contextual dissimilarity measure for accurate and efficient image search, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[42] H. Jegou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2011.

[43] J. Yuan, G. Gravier, S. Campion, X. Liu and H. Jegou, Efficient Mining of Repetitions in Large-Scale TV Streams with Product Quantization Hashing, Workshop on Web-scale Vision and Social Media, In *Proceedings of European Conference on Computer Vision (ECCV)*, 2012.

[44] D. Beaver, S. Kumar, H. C. Li, J. Sobel and P. Vajgel, Finding a needle in Haystack: Facebook's photo storage, In *Proceedings of the 9th USENIX conference on Operating systems design and implementation (OSDI)*, 2010.

[45] A. Bosch, A. Zisserman, and X. Munoz, Image classification using random forests and ferns, In *Proceedings of International Conference on Computer Vision (ICCV)*, 2007.

[46] T. Lindeberg, Feature detection with automatic scale selection, In *International Journal of Computer Vision (IJCV)*, 1998.

[47] H. Deng, W. Zhang, E. Mortensen, T. Dietterich and L. Shapiro, Principal Curvature-based Region Detector for Object Recognition, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[48] C. Harris and M. Stephens, A combined corner and edge detector, In *Proceedings of the 4th Alvey Vision Conference (AVC)*, 1988.

[49] S. M. Smith and J. M. Brady, SUSAN - a new approach to low level image processing, In *International Journal of Computer Vision (IJCV)*, 1997.

[50] E. Rosten and T. Drummond, Machine learning for high-speed corner detection, In *Proceedings of European Conference on Computer Vision (ECCV)*, 2006.

[51] M. A. Fischler and R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, In *Communications of the ACM (CACM)*, Volume 24 Issue 6, June 1981, Pages 381 - 395.

[52] A. Torralba, R. Fergus, and Y. Weiss, Small codes and large image databases for recognition, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[53] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, Large-scale image retrieval with compressed fisher vectors, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[54] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, Optimization by Simulated Annealing, In *Science*, 1983.

[55] R. H. Byrd, M. E. Hribar and J. Nocedal, An Interior Point Algorithm for Large-Scale Nonlinear Programming, SIAM Journal on Optimization, In *SIAM Journal on Optimization*, Vol 9, No. 4, pp. 877–900, 1999.

Summary

Learning Codeword Characteristics for Image Retrieval Using Very High Dimensional Bag-of-Words Representation

본 논문에서 우리는 초 고차원 Bag-of-Words (BoW) 표현법 기반의 이미지 검색을 위한 세 가지 문제를 다룬다. 세 가지 문제는 *tf-idf* 가중치 방법론과 계층적 스코어링 및 클래스 내부의 키워드에 관한 것이다. 대부분의 이미지 검색 시스템에서 사용하고 있는 *tf-idf* 가중치 방법론의 경우 일반적으로 이미지 검색의 검색성능을 향상시킨다고 여겨지지만, 실제로는 그 향상의 폭이 매우 작고 때로는 오히려 성능에 해를 끼치기도 한다. 계층 코드북 기반의 이미지 검색에 사용되는 계층적 스코어링의 경우도 스코어링에 고려하는 레벨의 수가 많아질수록 일반적으로 검색 성능이 향상된다고 여겨지지만, 데이터베이스의 성질에 따라 성능향상이 상이하고 특정 데이터베이스에 대해서는 오히려 성능에 해를 끼친다. 또한 대부분의 이미지 검색 시스템은 서로 다른 클래스는 서로 다른 키워드를 갖는다는 사실을 반영하지 않고 있다. 초 고차원 표현법 기반의 이미지 검색은 이미지 한 장이 초 고차원으로 기술되어 일반적인 기계학습 기법을 통하여 이러한 사실을 반영하는 것이 쉽지 않기 때문에, 대부분의 이미지 검색 시스템들은 모든 클래스에 대하여 동일한 하나의 가중치 벡터나 기준을 적용하고 있다.

우리는 이 세 가지 문제를 일관된 하나의 방법으로 해결하기 위하여 새로운 코드워드 가중치 방법론을 제안한다. 본 가중치 방법론은 기존 BoW 표현법이 갖는 코드워드별 독립성을 보존한다. *tf-idf* 가중치 방법론과 계층적 스코어링의 경우, 우리는 기존의 방법론들이 각각 사용하고 있는 정보인 코드워드의 문서 빈도와 레벨 외에 아무런 추가 정보를 사용하지 않고 이 방법론들의 성능을 향상시키는데 초점을 맞춘다. 코드워드의 문서 빈도와 레벨은 클래스 간 구별성과 밀접하게 연관되어있는 지표이기 때문에 우리는 코드워드별 이 두 가지 값을 중요 지표라 정의한다. 또한 클래스 내부의 키워드를 고려하는 문제의 경우, 클래스 내부에서 코드워드별 발생 빈도의 분산이 이들의 중요성과 밀접한 관련이 있기 때문에, 우리는 이러한 종류의 값을 역시 중요 지표라 정의한다. 또한 한 코드워드가 가지는 중요 지표 값으로부터 가중치 값으로 변환해주는 가중치 매핑 함수를 정의하였다. 주어진 데이터베이스에서 구별성을 최대화 시키는 최적의 가중치 매핑 함수 모양을 결정하기 위하여 우리는 주어진 데이터베이스로 부터 무작위로 표본을 추출하여 가중치 매핑 함수의 모양을 최적화 기법으로 학습하였다. 문서 빈도와 레벨을 가중치로 매핑하는 함수의 경우 전체 클래스에 동일하게 적용되는 전역 가중치 벡터를 생성하고, 클래스 내부의 분산을 가중치로 매핑하는 함수의 경우 각 클래스별로 적용되는 서로 다른 가중치 벡터들을 생성한다.

우리는 제안한 방법론을 검증하기 위하여, UKbench 데이터베이스와 Oxford5K 데이터베이스를 이용해 성능을 평가하였다. 본 방법론은 온라인 상태에서 추가되는 계산 비용은 거의 없었으며, 코드워드의 문서 빈도와 레벨 외에 아무런 추가 정보 없이 기존의 *tf-idf* 가중치 방법론과 계층적 스코어링 방법을 능가하였다. 또한 우리가 제안한 클래스별 키워드를 고려하는 가중치 방법론의 경우도 이를 고려하지 않은 검색 성능을 크게 능가하였다.

감 사 의 글

저는 국내 최고의 연구 중심 대학 KAIST의 전기 및 전자공학과에서, 컴퓨터 비전 분야의 권위자이신 권인소 교수님의 지도아래, 세계 유수의 연구진들이 주목하는 논문들을 쉼 없이 출판하며 세계와 어깨를 나란히 하고 계신 실험실 선배님들과 동료들의 도움을 받으며 공부하고 연구하였습니다. 이런 교수님의 기대와 선배님들의 연구업적에 비하면 부족한 점이 너무도 많은 연구지만, 마침내 제가 무언가를 만들어 냈고 그 결과를 평가받아, 공학석사 학위 취득을 앞두고 있습니다. 제가 살아오는 동안 한 차례도 바뀌지 않은 과학자가 되겠다던 꿈을 이루기 위해서는 넘어야 할 산들이 아직 너무 많지만, 막연하기만 했던 과학자들이 하는 그 무언가를 2 년의 시간동안 해보았고 그를 평가받았다는 사실만으로 저 스스로 가슴이 벅차고 뿌듯합니다. 제가 이렇게 좋은 환경에서 맺은 작은 결실은 결코 저의 능력이 아니라, 주변에서 저를 도와주신 많은 분들이 있었기 때문에 가능하였습니다.

또한 우리 실험실 구성원이 되기에 미달인 성적에도 저를 받아주시고, 그럼에도 하고 싶은 연구를 시켜주시고, 항상 강한 동기를 주시며 부족한 연구를 지도해주시고 도와주신 교수님께 깊은 감사의 말씀을 드립니다. 연구뿐 아니라 새로운 도전을 해보려는 저를 혹시 연구에 방해가 되지 않을까 걱정하시기에 앞서 그 도전을 바탕으로 한 강한 동기가 연구에 시너지가 되도록 조언해 주시고 지도해 주셨습니다. 이처럼 강한 동기를 바탕으로 한 도전적 연구자의 모습을 강조하시는 교수님의 말씀은 앞으로도 과학이란 범주 안에서 살아갈 저에게 가장 큰 지표가 되었습니다. 또한 바쁘신 일정에도 저를 심사해 주시고 제 연구에 대한 주옥같은 조언을 주신 김성대 교수님과 유창동 교수님께도 감사드립니다. 더 말할 나위 없이 세상에서 누구보다 헌신적인 저의 부모님은 저를 믿고 저의 생각을 항상 존중해주시며 아낌없이 지원하고 이렇게 키워주셨습니다. 유년 시절부터 다양한 자연·과학 체험을 누구보다도 많이 시켜주신 부모님은, 저를 실력을 떠나 과학의 범주에서 생각하고 답을 얻어내는 과정을 흥미로워하고 즐거워하도록 인도하셨습니다. 재차 강조해도 모자랄 만큼 저의 부모님께 가장 큰 감사를 드리며, 최근 이세를 가진 형에게도 감사의 말을 전합니다.

항상 밤낮이 뒤바뀐 채로 살며 실험실 지각의 주범인 저는 지난 2 년간 실험실 구성원들에게 큰 누를 끼친 것 같아 미안한 마음이 큽니다. 야행성 생활이 학부시절 부터 몸에 습관이 된지라 아직도 잘 고쳐지지 않지만, 실제로 많이 노력하고 있고 앞으로는 더 좋은 모습 보여드리도록 노력하겠습니다. 이런 저에게도 큰 꾸짖음 없이 격려해주시고 성심껏 도와주신 실험실 선배님들과 동료 후배들에게 감사드립니다. 특히 저와 과제를 같이 하고 있는 유경 선배께 좋은 모습보다는 실수하는 모습을 자주 보여드렸는데도 항상 웃음으로 대해주시고 과제가 잘 마무리 되도록 리드하시는 모습을 보면서 고마움도 느끼고 많이 배웠습니다. 신입생 첫 서울 출장에서 유경 선배를 수 시간 기다리게 한 제 불찰에 늘 죄송한 마음을 갖고 있습니다. 이미 박사 졸업을 하셨음에도 후배들의 연구 뿐 아니라 잡스러운 일까지도 성심 성의껏 도와주시는 복윤수 박사님, 무서운 드라이빙으로 신세계를 경험시켜주시고 가끔씩 맛있는 것들을 사주시는 재훈 선배, 아직 이야기를 많이 나누어보지 못해 아쉬운 가운 선배와 영근 선배 성제 선배, 묵묵한 카리스마를 가지고 계시면서도 가끔씩 따뜻한 안부를 물어봐 주시는 김정호 박사님, ACCV 행사 때 박스위에서

자고 있는 저에게 말없이 외투를 덮어주시고 나가신 록헌 선배, 제 연구의 의문점들을 조근 조근한 목소리로 친절하게 해결해 주시고 우리 과제에 가장 큰 공헌을 남겨주신 채훈 선배, 얼마 전에도 제 연구에 주옥같은 조언을 해주시고 또 물어볼게 많이 있는 형우 선배께도 깊은 감사의 말씀 전합니다. 부족한 저를 뽑는데 가장 큰 역할을 해주시고 저와 애플빠로 공감하는 동걸 선배, 랩장의 역할을 헌신적으로 하느라 바쁘심에도 제 연구에 깊은 논의를 해주시고 실제로 결과 도출에 큰 도움을 주신 준영선배께도 너무 감사드립니다. 우리 실험실에서 제일 엉뚱한 웃음을 주시고 항상 친절하신 천사 한화 팬 영진 선배, 우리 랩에 들어오는데 미리부터 도움을 주고 항상 친절한 안내를 해주며 디펫이란 인맥을 공유하고 있는 지영이, 항상 묵묵히 연구하시다 갑자기 귀여운 개그를 선사하시는 한화 팬 재식선배, 영진 선배의 사랑을 독차지하고 계신 항상 친절한 승학 선배, 항상 말 걸어 주시면서도 짓궂은 장난으로 저를 남감하게 하시는 인욱 선배, 실험실에서 제일 친근하고 편안한 형 병태 선배께도 감사드립니다. 어느새 박사 과정에 올라가 놀랄 만큼 멋진 연구 하고 있는 태현이, 조용하게 묵묵히 연구하는 보더 효원이, 이번 석사 졸업 준비하느라 같이 너무 고생했고 위안이 되어준 우리 동기 용섭이와 유덕이 에게도 고마운 마음이 큽니다. 특히 유덕이는 스키장 안 갔다고 삐진 것이 이제 풀린 것 같아 다행으로 생각하고, 용섭이는 또 다시 박사 동기가 되어 서로 의지하며 연구하기를 기대하고 있습니다. 항상 저에게 부담스러울 정도의 과찬을 해주시고 가끔 음료수를 마시며 딥토킹을 나누는 고마운 한화 팬 해곤이형, 우리 랩에서 개인적으로 제일 웃기다고 생각하고 동갑이여서 더 정감이 가는 상우, 랩에 들어와 궂은일 마다하지 않고 많은 일들을 하면서도 뛰어난 연구를 준비하고 있는 신입생 순민, 경돈, 종원, 민정, 경민이 에게도 고마움을 전합니다. 특히 결국 내지 못한 논문을 도와주느라 같이 밤새며 고생해준 순민이 에게는 더 큰 고마움을 전합니다. 이 외에도 사회에서 멋진 모습으로 연구하고 계신 정예근 박사님과 원진 선배, 순기에게도 감사드립니다. 특히 정예근 박사님은 아무것도 모르고 연구하던 제게 큰 용기를 주고 방향을 잡아주시는 조언을 해주신 점 감사드리고 원진이형은 정말 친구처럼 친하게 지내고 자주 놀러갔던 추억이 그립습니다. 또한 엄청난 양의 일들을 탁월한 능력으로 문제없이 처리해 주시는 지은 누님이 없었다면, 저의 과제도 혼란이 많았을 것입니다. 감사합니다.

내 논문 도와주겠다며 자신의 숙제도 못 내고 같이 밤새면서 고생해주고 실제로 많은 도움을 준 환순이 희선이, 멋진 미래를 위해 큰 야망을 품고 기술에 대한 도전을 함께하고 있는 cloudesigners의 수장이자 저의 태양 디펫형, 우리 팀의 따뜻한 엄마이자 엄청난 능력자 알엠형, 재수없지만 돈이 많고 정도 많이 들어서 친하게 지낼 수밖에 없는 크롤러 큐, 저와 큐이 무서워하고 한 살 어린 동생이지만 남이 보면 형인 무서운 형 팽, 그리고 최근 합류한 오랜 소울 메이트이자 믿음직스러운 비즈니스맨 봉구에게도 깊은 고마움을 표합니다. 또한 제 학부 시절 소중한 추억의 대부분을 차지하고 있고 저의 대학생활을 수렁에 빠뜨린 흑인 음악 동아리 구토스의 형제들 란뚜, 메치, 병국, 조철, 예강, 지훈, 영규, 민재, 홍민이에게도 고맙고 사랑한다는 말 전합니다. 특히 함께한 추억이 정말 많은 메치는 술을 제발 그만 마신다면 더욱 고마울 것 같고 란뚜는 연락 좀 한다면 더욱 고마울 것 같습니다. 각자 바쁜 일정에도 학교로 자주 찾아와준 보문고 동창인 약사님 봉수와 이제는 대학도 동창이 된 민이에게도 고맙게 생각합니다.

이밖에도 언급하지 못한 친구들, 제가 아는 모든 인연에게도 감사의 말씀 전합니다. 이 모든 분들에게 저의 작은 결실이 조금이라도 기쁨이 되었으면 좋겠습니다. 앞으로 보다 어른스럽고 책임감 있는 모습, 기대에 부응하는 모습 보이도록 노력하겠습니다.

이 력 서

이 름 :	유 동 근
생 년 월 일 :	1986년 8월 13일
주 소 :	대전광역서 서구 둔산3동 국화아파트 204동 708호
E-mail 주 소 :	dgyoo@rcv.kaist.ac.kr

학 력

2002. 3. – 2005. 2.	대전 보문고등학교
2006. 3. – 2011. 2.	KAIST 전기 및 전자공학과 (B.S.)

경 력

2012. 2. – 2013. 2.	KAIST 전기 및 전자공학과 일반조교
2012. 4. – 2012. 12.	지식경제부 주관 창의 도전형 SW R&D 참여

학 회 활 동

1. **Donggeun Yoo**, C. Park, Y. Choi, I. Kweon *Intra-class Key Feature Weighting Method for Vocabulary Tree Based Image Retrieval*, The 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2012), November 2012.
2. **Donggeun Yoo**, C. Park, Y. Choi, I. Kweon *Image Retrieval by Important Feature Weighting for Each Class*, The 37th Korea Information Processing Society (KIPS) Spring Conference 2012 , April 2012.