Query Expansion for Visual Search Using Data Mining Approach

Author:
Siriwat Kasamwattanarote

Supervisor:
Prof. Shin'ichi Satoh

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

February 2016

A dissertation submitted to Department of Informatics,
School of Multidisciplinary Sciences,
SOKENDAI (The Graduate University for Advanced Studies),
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Advisory Committee

1. Prof. Shin'ichi Satoh	National Institute of Informatics, University of Tokyo
2. Assoc.Prof. Duy-Dinh Le	National Institute of Informatics
3. Prof. Akihiro Sugimoto	National Institute of Informatics
4. Prof. Imari Sato	National Institute of Informatics
5. Assoc.Prof. Gene Chung	National Institute of Informatics

"A human being is a part of the whole called by us universe, a part limited in time and space. He experiences himself, his thoughts and feeling as something separated from the rest, a kind of optical delusion of his consciousness. This delusion is a kind of prison for us, restricting us to our personal desires and to affection for a few persons nearest to us. Our task must be to free ourselves from this prison by widening our circle of compassion to embrace all living creatures and the whole of nature in its beauty."

— Albert Einstein

Acknowledgements

I wish to thank to whoever start exploring on the long journey of my thesis. This thesis is aimed for persuing deeper on my skill sets which will appear here for readers in term of information retrieval, image processing, data mining, and also software engineering.

I would like to offer my main supervisor, Prof. Shin'ichi Satoh, on his great assistance, support, and patient guidance along this hill climbing Ph.D. path. His willingness to give his time so generously has been very much appreciated. I would also like to thank for Assoc.Prof. Le Duy-Dinh for his advice and useful critiques that keep me improving my research work and presentation skill. My grateful thanks are also extended to the rest committees for their comments and suggestions that guided me to the right direction of what audiences really need. My special thanks are also for Dr. Naoki Chiba, who allowed me to join Rakuten Institute of Technology as an internship student. During that period, I can push my concentrate on the core part of this work.

I would also like to thanks for the following friends during my life in NII: Cai-Zhi Zhu, who let me coordinate along his researches to understand how the state-of-the-arts image retrieval work. Sebastien Poullot, who gave me hints and discussions on how to do image retrieval in practical ways. Merlin Zhang, who works mostly on Java, but rather pushed me a lot to convert all my C++ codes from an STL container based to a pure pointer based. Pannawit Samatthiyadikun, who connects a missing link on my knowledge of Galois connection, then speeding up the mining process is possible. Rathachai Chawuthai, who guided me look at the most important mistake on my research story that I really missed and gave the other people aspect on writing the academic papers. Also, thanks to my friends and all people around me, who are important and valuable, that shaped me be as today.

Lastly, thanks to the educational and financial support from National Institute of Informatics and SOKENDAI (The Graduate University for Advanced Studies) during my study. I appreciate all the incredible opportunities and memorable experience they offered.

I wish to give the most thankful to my family, who always support me in any decisions, who guides and gives me the answer when I really need it, and Lee Hyunju, who always beside me while I was in the most blueness time. *And the best thanks for my parents, who are truly my everything.*

Siriwat K.

1 January 2016

SOKENDAI (THE GRADUATE UNIVERSITY FOR ADVANCED STUDIES)

Abstract

School of Multidisciplinary Sciences
Department of Informatics

Doctor of Philosophy

Query Expansion for Visual Search Using Data Mining Approach

by Siriwat KASAMWATTANAROTE

Bag of Visual Words (BoVW) framework is known to be effective for image retrieval. Query expansion (QE) further boosts the retrieval performance by refining query with relevant visual words which are geometrically consistent between query image andtc each of highly ranked retrieved images obtained from the first round retrieval. Since QE checks pairwise consistency between query and highly ranked images, the performance may be deteriorated by slight degradation of query image. We propose *Query Bootstrapping* (QB) as a variant of QE to circumvent this problem by using consistency among highly ranked images instead of pairwise consistency.

In so doing, we regard frequently co-occurring visual words in highly ranked images as relevant visual words. Frequent itemset mining (FIM) is used to find such visual words efficiently. However, the FIM-based approach requires sensitive parameters to be fine-tuned, namely, support (min/max-support) and the number of top ranked images (top-k). Here, we propose an adaptive *support* algorithm that adaptively determines both the minimum support and maximum support by referring to the first round's retrieval list. Selecting relevant images by using a geometric consistency check further boosts retrieval performance by reducing outlier images from a mining process. An important parameter for the LO-RANSAC algorithm that is used for the geometric consistency check, namely, *inlier threshold*, is automatically determined by our algorithm. We further introduce *tf-fi-idf* on top of *tf-idf* in order to take into account the frequency of inliers (*fi*) in the retrieved images.

We evaluated the performance of QB in terms of mean average precision (mAP) on three benchmark datasets and found that it gave significant performance boosts of 5.37%, 9.65%, and 8.52% over that of state-of-the-art QE on Oxford 5k, Oxford 105k, and Paris 6k, respectively. Also, we extend the datasets with MIRFlickr 1M distractor and tested with several severed query conditions to prove our robustness achievements.

Contents

List of Figures

List of Tables

Abbreviations

IR	**I**nformation **R**etrieval
CBIR	**C**ontent-**B**ased **I**nformation **R**etrieval
AQE	**A**verage **Q**uery **E**xpansion
BoW	**B**ag **o**f **W**ord
BoVW	**B**ag **o**f **V**isual **W**ord
QE	**Q**uery **E**xpansion
QB	**Q**uery **B**ootstrapping
SP	**S**patial **V**erfication
RANSAC	**Ran**dom **Sa**mple **C**onsensus
LO-RANSAC	**L**ocally **O**ptimized **RANSAC**
SIFT	**S**cale-**i**nvariant **F**eature **T**ransform
DOG	**D**ifference **O**f **G**aussian
GLOH	**G**radient **L**ocation and **O**rientation **H**istogram
MSER	**M**aximally **S**table **E**xternal **R**egions
SURF	**S**peeded **U**p **R**obust **F**eatures
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
BRISK	**B**inary **R**obust **I**nvariant **S**calable **K**eypoints
ORB	**O**riented **B**RIEF
ASIFT	**A**ffine-**S**IFT
OpenCV	**O**pen **C**omputer **V**ision
FIM	**F**requent **I**temset **M**ining
LCM	**L**inear time **C**losed itemset **M**iner
KDTree	**K**-**D**imensional **T**ree
MVS	**M**obile **V**isual **S**earch
FLANN	**F**ast **L**ibrary for **A**pproximate **N**earest **N**eighbors

FINT	**F**ixed **I**nlier **T**hreshold
ADINT	**A**daptive **I**nlier **T**hreshold
AP	**A**verage **P**recision
mAP	**m**ean **A**verage **P**recision
CPU	**C**entral **P**rocessing **U**nit
RAM	**R**andom **A**ccess **M**emory
GB	**G**iga **B**yte
TB	**T**era **B**yte
RHEL	**R**ed **H**at **E**nterprise **L**inux
GCC	**G**NU **C**ompiler **C**ollection
PCA	**P**rincipal **C**omponents textbfAnalysis
TREC	**T**ext **RE**trieval **C**onference
TRECVID	**TREC V**ideo **R**etrieval **E**valuation
MFU	**M**ost **F**requently **U**sed
SSD	**S**olid-**s**tate **D**rive
HDD	**H**ard **D**isk **D**rive
INV	**INV**erted index file
OS	**O**perating **S**ystem
HE	**H**amming **E**mbedding
GVP	**G**eometric **V**isual **P**hrases
WGC	**W**eak **G**eometric **C**onsistency

Dedicated to my family

Chapter 1

Introduction

"If there's a book that you want to read, but it hasn't been written yet, then you must write it."
— Toni Morrison

FIGURE 1.1: The images taken as memorable memories during my study in Japan.

1.1 The motivation of this research

In the recent days of the information era, images were taken a lot by using several handy devices such as a smart phone and a digital camera. These devices create a kind of time-memory as the moments of one people life (e.g., figure 1.1), which will be used for reminding back their memory in the previous days. As the current social media and social network technologies allow people to connect and share their life and memories with friends, million images were rapidly created and increasing size of valuable image collection these days, which are being retrieved back to the devices for a looking back purpose. The aforementioned flow can be seen as in figure 1.2.

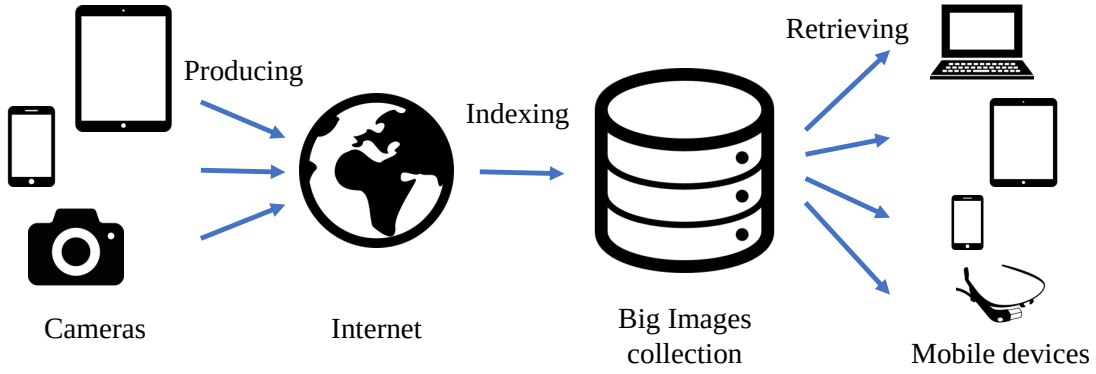


FIGURE 1.2: Figure illustrate how images were created and retrieved in the this era.

In case we use any services for sharing our memories from several devices like be a PC, a mobile phone, a tablet, or even on a wearable device through the web services (e.g., Instagram, Facebook, Twitter, Flickr, etc.). Retrieving back any image is meant to load a particular image directly from the server to visualize it on your screen by using a direct URL, hash tags, or any provided API [1, 2]. Additionally, images are being indexed into collections of big image databases with specific meta-data, hash tag, time stamp, geographical location, or even a specific ID. This kind of indexing is to map the unstructured data into a high-level structured space. And the retrieval will be fast and easy just like following look up tables on a server for an exact physical location of the recalling image. On the other hand, searching for an object within one or multiple images is hard and it will be even more complicated if the image was not provided with any specific meta-data. For this reason, indexing cannot be done by mapping image to high-level structured data, but rather be performed by more sophisticated techniques calling as bag-of-visual-word (BoVW) framework (see section 1.2).

In this research, we focus on the way to search for any specific object from a big image collection, which cannot be stored on a mobile devices or even a PC. So, we conduct our research based on a client-server architecture for handling a large-scale image database and for simulating the way people search for an object while they usually use their own mobile device. However, the problems raised to researchers and engineers in order to serve the end-user fast and accurate under several constraints, for instance, a database size, an image quality, etc.

1.2 Background

1.2.1 Information retrieval

In the early year of information retrieval (IR) when people begin indexing and searching for texts in the documents, a document may contains several sessions. One session may also contains many paragraphs. Several sentences together form up one paragraph. And yes, the smallest portion of a document that we may want to find is a word or multiples that connecting together for building up a sentence. Searching for a specific word from one documents might be easy by looking up to an index (figure 1.3) at the end of a document (i.e., a book). In contrast for a real situation, we do not even know a book name, when the information retrieval technology allows, it helps us by collecting up an index from the several books together which made us easy in searching on one index of the whole book collection. This technique is called an inverted indexing [3], which is widely used in several successful approaches of the information retrieval applications nowadays. The underlying search algorithm for this scenario is early adopted on a standard boolean model [4] with query representation as a set of words as called as bag-of-word (BoW) [3] recently. Namely, each document is regarded as an unordered collection (a "bag") of words and represented as N-dimensional histogram of word occurrences, where N is the number of words in a relies language. Hence, this model tells only the existing of word appearance in a document that works on a very small document, which is called a boolean model, that stores a term appearance in binary.

Common usages of text search are not only depend on one word, but rather refer to a phrase or multiple words grouped together as a query. Then, a vector space model as proposed in Salton et al. [5] has more advantages over the standard boolean model because the matching result of words similarity is more informative when it not only points out the existing of word appearance but also explains how much each word has been matched in the documents. However, some words, like "*the*" and "*and*", usually

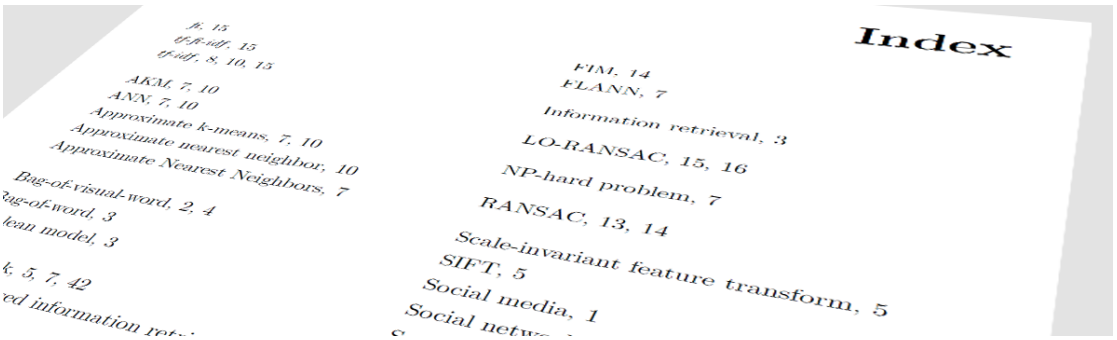


FIGURE 1.3: The actual index appeared on a book helps readers find an appropriate page for an interesting word or a phrase.

appear more often than the others, though they are less informative than infrequent words. To handle this kind of "stop word" with extremely high frequency, the words in the vector space model are weighted based on how informative they are.

A term weighting *tf-idf* of Salton and Buckley [6] is then introduced for emphasizing a weight of rarely found words, while reducing a weight of such "stop word" or any words that appear several time in several documents. The first term *tf* is a *term frequency weighting* as the value of each histogram bin is equal to the number of times the word appears in the document. Another tern *idf* is the *inverse document frequency* as $log(1+\frac{1}{N_i})$, where N_i is the number of documents in which word i appears. The overall bag-of-words representation is then weighted by multiplying the *term frequency tf* with the *inverse document frequency idf* as a final product of the *tf-idf* weighting as in Salton and Buckley [6].

Thereby, similarity calculation between queries and documents is computed using cosine similarity between *tf-idf* weighted bag-of-word histograms, and normalizes into a document length. This yields documents to be sortable by its score between the given query and itself.

1.2.2 Image retrieval

Similarly to visual contents, content-based information retrieval (CBIR) is known as query by image content or an image retrieval by using computer vision techniques to the information retrieval problem. This technique is rather means to search by analyzing the contents of an image rather than the keywords, specific tags, meta-data, etc. The term of contents is comparable to words in a text documents. However, the content definitions are derived from the image itself, without having humans manually annotate images by entering keywords or meta-data. This is due to the problem can far larger from human to do it within a small laboratory group, However, with a lot of human-hour, [7] contributes the ImageNet as the image annotations organized according to the WordNet lexical hierarchy Miller [8]. In face, ImageNet is formed just for a high level human groundtruth.

For a extracting low-level contents within an image, the Bag of Visual Words (BoVW) framework has been an effective means of an object-based image retrieval since it was first described in Video Google work of Sivic and Zisserman [9], but still very popular, and later in a work [10]. A definition of the BoVW can be in a form of the histogram representation based on independent of visual words. To represent an image using BoVW histogram, an image can be treated as a document. Similarly, "words" in images need to be defined too. To achieve this process, it usually includes following three steps:

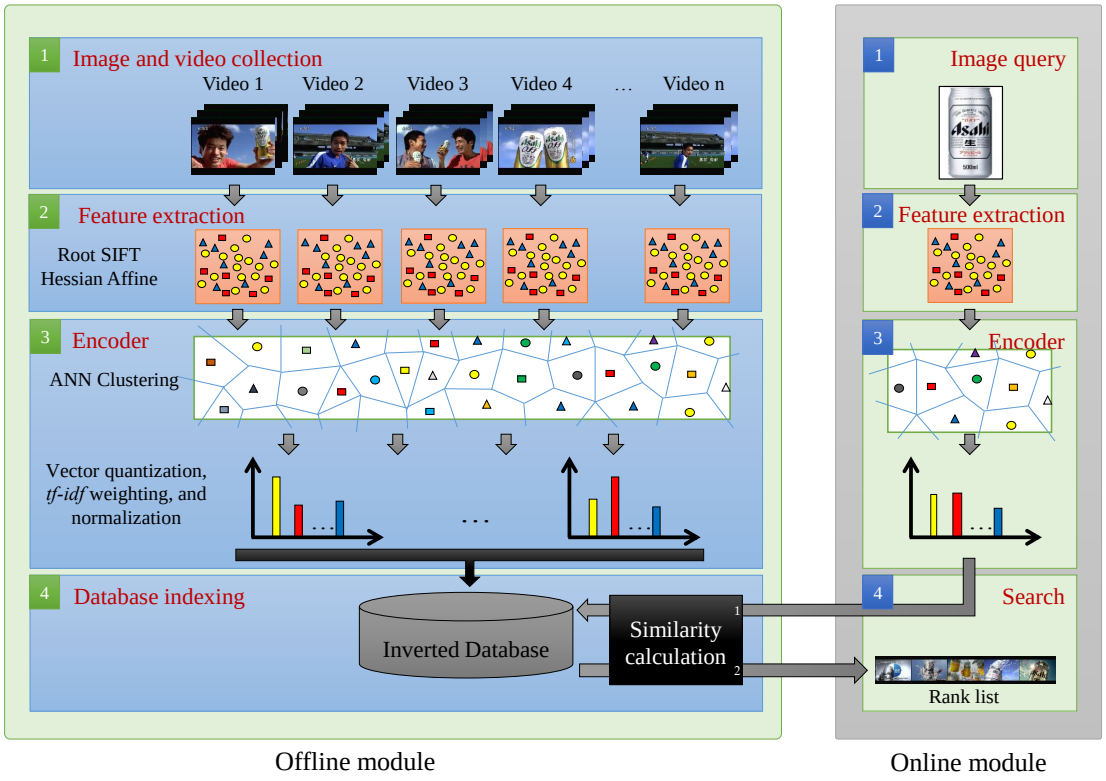


FIGURE 1.4: The architecture shows how images and videos were handled during the database indexing process (offline module) and how the query image was handled during the retrieval process (online module). The final product of both online and offline modules is a BoVW histogram, which can be calculated the similarity matching for ranking the final result.

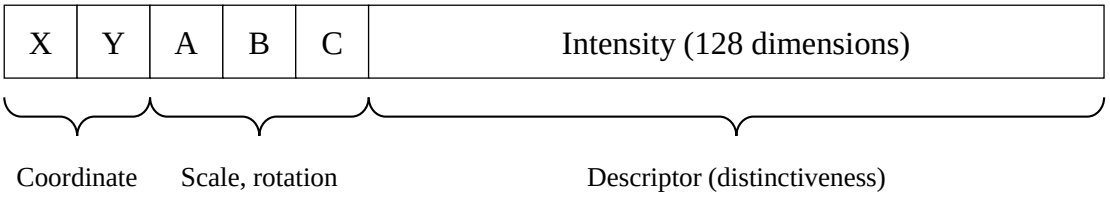


FIGURE 1.5: The data structure shows how SIFT with Hessian Affine store a data for each keypoint. X,Y are the coordinate of a keypoint within an image space. A,B,C is the values describe how affine appears around a keypoint. Intensity is the descriptiveness around a keypoint that consisted with 128 dimension for each keypoint.

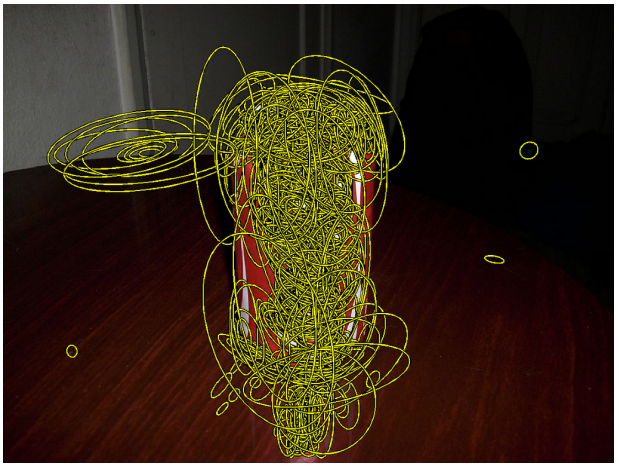
feature extraction (feature detection and feature description), codebook generation , and database indexing as shown in figure 1.4 which represents an architecture of image and video retrieval scenario using BoVW baseline.

1.2.2.1 Feature extraction

Start from a feature extraction, Lowe [11] proposed a scale-invariant feature transform (SIFT) . Later, SIFT is the most famous image feature that is widely used in most

(A) An original image.

(B) An image with the overlayed SIFT keypoints in green dots.



(C) An image with the overlayed SIFT keypoints with Hessian Affine detector produced affine region in yellow ellipses.

FIGURE 1.6: The figures show (A) Original image sample. (B) Overlayed image of keypoints detected by using a blob detection originally used in SIFT feature detector. (C) Overlayed image of affine regions detected by using Hessian Affine detector.

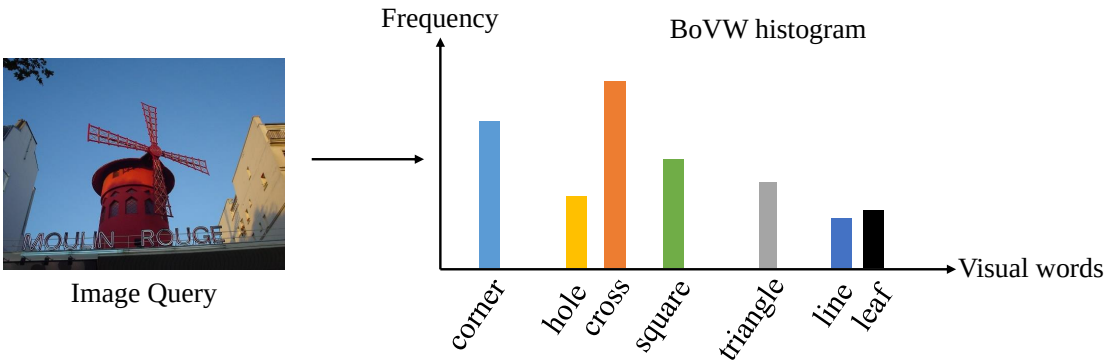


FIGURE 1.7: The illustration shows an image was transform into BoVW histogram, which represented by a frequency of visual words. Visual codebook is similar to a text dictionary, by comparing, visual word may consisted of a corner, hole, cross, square, triangle, line, leaf, etc.

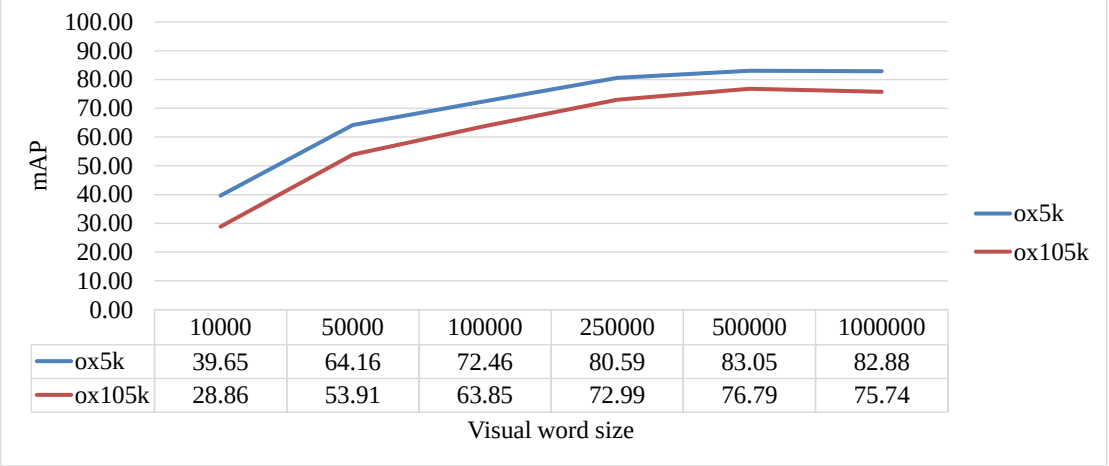
of visual related applications. The original SIFT package comes with both feature extraction and feature description. The feature extraction is to detect points of interest, which is called as keypoints in the SIFT framework, as the spatial information (x, y coordinate) as seen in a figure 1.6b. The feature description is to describe the keypoints such that the descriptor is highly distinctive depends on the illumination appeared on each keypoint. In this step, SIFT generates 128 dimensions for each keypoint as its descriptor. Since the characteristic of general keypoint detector using blob detection is not robust to the affine change, The works named Hessian Affine region detector of Mikolajczyk and Schmid [12, 13], Mikolajczyk et al. [14] come to solve this problem and then boost the performance of SIFT when it was employed in the works of Michal Perďoch [15, 16], which generate an affine region information as an ellipse (A,B,C) according to the follows:

$$A(X-U)^2 + 2B(X-U)(Y-V) + C(Y-V)^2 = 1 \tag{1.1}$$

where the rotation in degree (Θ) can be obtained as follows:

$$\Theta = \arctan\left(\frac{B}{A-C}\right) + \frac{\pi}{2} \tag{1.2}$$

To understand more on SIFT feature, the figure 1.5 explains how SIFT Hessian Affine store the data for each keypoint. Finally, the result of Hessian Affine keypoints can be seen in figure 1.6c as the ellipses illustrate an affine with a scale and a rotation from SIFT.



	10000	50000	100000	250000	500000	1000000
ox5k	39.65	64.16	72.46	80.59	83.05	82.88
ox105k	28.86	53.91	63.85	72.99	76.79	75.74

FIGURE 1.8: The experiment for tuning a visual vocabulary size for two standard dataset, namely, Oxford 5k and Oxford 105k. The result shows the same aspect as increasing a vocabulary size will improve a retrieval performance.

1.2.2.2 Codebook generation

In order to define image contents as words for image, namely, visual word (VW), the image feature is used in this purpose. According to a SIFT structure (figure 1.5), the descriptor portion as 128 dimensions vector is descriptive enough to be assigned as a word. The codebook generation step is to convert vector-represented image patches to visual codewords (similar to words in text documents), which also produces a visual codebook (similar to a word dictionary). One simple method is performing clustering technique like k-means [17, 18] over all the descriptor vectors [19, 20]. However, the traditional k-means is simply an NP-hard problem [21] for a general vector dimensions and a number of clusters or a vocabulary size, which we prefer to use 1 million visual words (as our experiments in figure 1.8) through all our experiments.

In handling this, Philbin et al. [22] proposes to use an approximate k-means method (AKM) [23] to reduce the time usage, while increasing performance by the a vocabulary size. Thus, each descriptor in an image is mapped to a certain visual codeword id through the quantization process. By assigning codeword to nearest cluster using a nearest neighbor search algorithm. Also, the approximate nearest neighbors (ANN) is also help speedup the overall process (e.g., Fast Library for Approximate Nearest Neighbors (FLANN) of Muja and Lowe [24], Muja [25]) Finally, an image can be represented by the histogram of local descriptors assigned as codewords, namely, visual words to produce the final product of a BoVW framework called a BoVW histogram (see figure 1.7). Additionally, BoVW histogram can be sparsely filled with non-zero words when the total number of keypoint is far smaller than the cluster size, i.e., dictionary size. A good vocabulary size for data is also been studied in this work [26].

1.2.2.3 Database indexing

Assuming we have 1000 images consisted with 1000 keypoints each, the total number of visual words to be stored in a database is 1 million words. In a very naive way like bruth force, searching for a query image with 10 keypoints will need to load all 1000 images and find for any of matched words, in total, 1 million words need to be loaded for calculating a similarity between a query image and the all images in the database. This technique will be suffered from a larger database size. The technique called inverted index [3, 6] comes to solve the mentioned problem. Inverted indexing technique is a one of the most successful component of a typical search engine and indexing algorithm. A goal of this technique is to optimize the speed of a query, by finding the documents where word "X" appeared. The time, memory, and processing resources to perform such a query are saved as seen in table 1.1. With the inverted index created, the query can now be resolved by jumping to the word id in the inverted index.

However, the existence of irrelevant visual words may degrade the performance of image retrieval based on BoVW. Here, a word-weighting schemes such as *tf-idf* inspired by the idea of text retrieval [3, 6], have been proposed in order to reduce the effect of irrelevant visual words, which also applicable to a visual based information retrieval like BoVW.

Document	Word	Total
1	cat,fish,sea,tree	4
2	egg,bread	2
3	bicycle,tree,stone,sea	4
4	fish,bread,water	3
5	cat,stone	2
6	cat,dog,mouse,cheese	4
7	bread,fish,water	3
8	tree,bicycle	2
9	egg,mouse,fish	3
10	cat,tree,fish	3

Query = {fish,stone,sea,water}
Disk/memory seek: **7 times** (for documents)
Load: **22 memory slots** from doc = {1,3,4,5,7,9,10}

(A) A normal database.

Word	Document	Total
bicycle	3,8	2
bread	2,4,7	3
cat	1,5,6,10	3
cheese	6	1
dog	6	1
egg	2,9	2
fish	1,4,7,9,10	4
mouse	6,9	2
sea	1,3	2
stone	3,5	2
tree	1,3,8,10	3
water	4,7	2

Query = {fish,stone,sea,water}
Disk/memory seek: **4 times** (for words)
Load: **10 memory slots** from word = {fish,stone,sea,water}

(B) An inverted indexing database.

TABLE 1.1: The memory usage comparison of a toy example between a normal database (left) and an inverted index database (right).

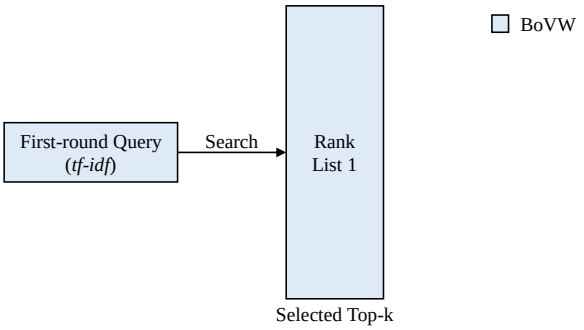


FIGURE 1.9: A framework overview of a standard back-of-visual-word.

1.3 Preliminaries

In this section, we explain notations used throughout our research along with algorithms of baseline former approaches, namely, bag-of-visual-word (BoVW), query expansion (QE), average query expansion (AQE), and frequent item sets mining (FIM).

1.3.1 Object-based image retrieval (BoVW)

The main purpose of object-based image retrieval is to find the most similar objects to a given query image $\boldsymbol{Q}$ in a database $\boldsymbol{D}$ consisting of M images, where $\boldsymbol{D} = \{I_1, I_2, ..., I_M\}$. An object-based image retrieval refers as a basis of standard BoVW framework, which can be illustrated as a simple architecture shown in figure 1.9.

If a query is given without any specific bounding box around the intended object, the image search will interpret the whole image as being the query and will retrieve images with the globally highest similarities to it from the database. In some cases, the query is a region of interest (ROI) in an image. The query with ROI is meant to regard within only an ROI portion as a target object of retrieval as shown in figure 1.10.

Here, Eq. 1.3 describes the result of such a simple retrieval scheme $\boldsymbol{R}$ for a (normally ranked) list of images.

$$\boldsymbol{R} = \{I_b \in D | I_b \text{ contains object appeared on } \boldsymbol{Q}\} \tag{1.3}$$

The goal of object-based image retrieval, however, is not to identify the most similar image, but rather to find images from the database collection that contain the same or visually similar objects, as in the Trecvid Instance Search task [27, 28].

FIGURE 1.10: An example of object-based image retrieval using ROI as a specified object on top of a query image $\boldsymbol{Q}$. Retrieval is done on an inverted index database $\boldsymbol{D}$. A rank list $\boldsymbol{R}$ is a similarity result ordered by a score between $\boldsymbol{Q}$ and each of matched image in $\boldsymbol{D}$.

We chose BoVW as the baseline for comparison in this study. BoVW extracts local features from each image (i.e., from the query and the images in the database). In particular, we used a Hessian Affine keypoint detector by Michal Perďoch [15] and RootSIFT by Lowe [29], Arandjelovic [30] to obtain on average 1,000 to 3,000 visual descriptors per image, which is varies depends on the content of an image. We used approximate k-means (AKM) by Philbin et al. [22, 23] to generate a large visual vocabulary ($K = 1$ million) (see figure 1.8) and approximate nearest neighbor search (ANN) by Muja and Lowe [24], Muja [25] to quantize the local features for the sake of speed. A k-dimensional vector was obtained for each image. Here, we denote the vector of the query (optionally with the ROI) as Q and the vector of the database image as I_i. Moreover, the vector components can optionally be weighted by *tf-idf* . We used the following similarity between Q and I:

$$sim(\boldsymbol{Q}, I) = 1 - \left\| \frac{\boldsymbol{Q}}{\|\boldsymbol{Q}\|_1} - \frac{I}{\|I\|_1} \right\|_1 \tag{1.4}$$

where $\|\cdot\|_1$ denotes the L^1-norm. This similarity was calculated from all matched non-zero words, or for only the matches inside the ROI, if provided. The top-k retrieved results $\boldsymbol{R}$ were those with the k highest similarities.

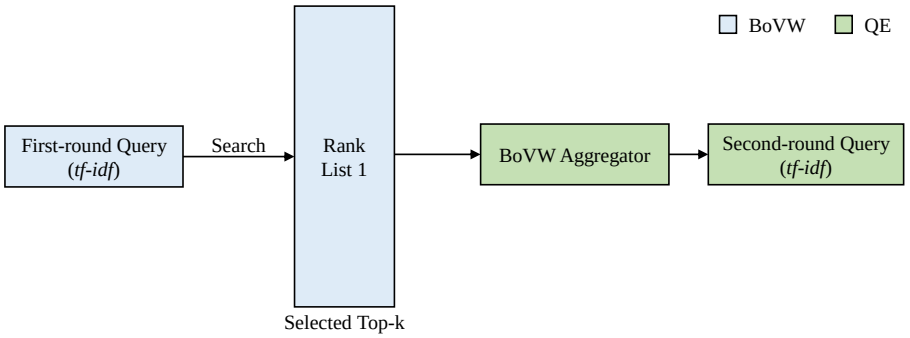


FIGURE 1.11: A framework overview of a standard query expansion (QE).

1.3.2 Query expansion (QE)

Given an image query, the retrieval algorithm returns a ranked list by checking the similarities between the query image and the images in the database. After the first round of image retrieval, query expansion (QE) is a well-known technique that boosts the performance in the field of text-based information retrieval. It reformulates a new query from information found in highly ranked documents. The architecture of QE is show in figure 1.11. QE is regarded as a technique to improve the total retrieval recall by adding terms that were not found in the original query but might be found in the retrieved documents.

For image retrieval, given an initial top-k ranked list $\boldsymbol{R}$, QE assumes that (most of) the images in the list are relevant and generates a next-round query by averaging the BoVW histograms of the images retrieved in the first round.:

$$\boldsymbol{Q'} = \frac{\sum_{b=1}^{k} \boldsymbol{R}_b}{k} \tag{1.5}$$

where $\boldsymbol{R}_b$ denotes the BoVW histogram of the b-th image in $\boldsymbol{R}$.

Since the BoVW method is inspired by text-based information retrieval, it is quite natural for QE to be applied to image retrieval as well. One of the simplest QE techniques is called pseudo relevance feedback or blind relevance feedback [31]. The method feeds these relevant documents back to the system assuming that the top k of the retrieved ranked items are relevant and then generates the query by aggregating the features of these items. In object-based image retrieval, the standard way is to use the average of BoVW histograms of the top-k items as the refined query for the next round retrieval in order to get improved retrieval performance without an extended of user interaction. This technique works well in text-based information retrieval, since each term does not have a strict word order. In contrast, an image has a meaning that depends on the spatial locations of its visual words Hence, a simple application of QE to image retrieval

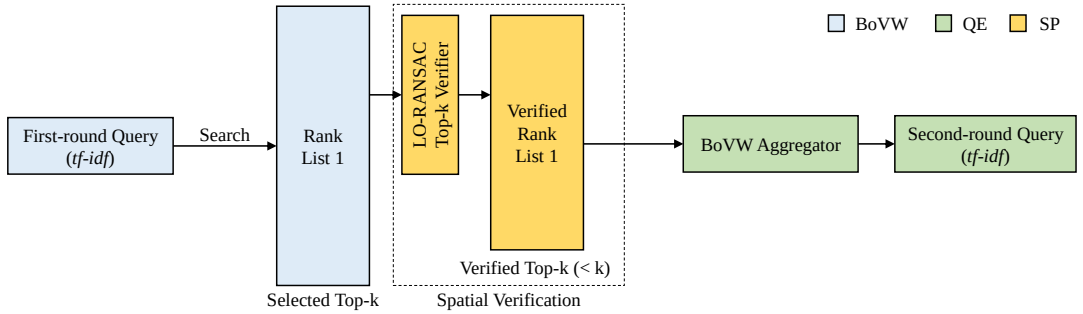


FIGURE 1.12: A framework overview of a standard average query expansion (AQE) with a spatial verification module (SP).

may fail because irrelevant images may be included among the highly ranked results and also because visual words in background regions may be included in the relevant images. Therefore, the retrieved images may be irrelevant to the query.

1.3.3 Average query expansion (AQE)

To circumvent the problem of background clutter as in QE, Chum et al. [32] takes geometric information into account in QE. In order to take the advantage of geometric topology information, Total Recall (Chum et al. [32, 32]) adapts spatial verification (using RANSAC or the like) between the query image (or query region) and each of the database images for an object-based image retrieval. Then the method is widely known as average query expansion (AQE). Given a ranked list of retrieved images according to the query image with query region, AQE first checks the geometric consistency between the query region and each of the retrieved images and only selects relevant visual words in verified images appearing in back-projected regions to be put into the next-round query. Here, let us assume that k' images successfully pass the geometric consistency check ($k' \leq k$) and $\boldsymbol{R}'_b$ is the BoVW histogram of the b-th image. The next-round query of AQE is computed as:

$$\boldsymbol{Q}'' = \frac{\boldsymbol{Q} + \sum_{b=1}^{k} \boldsymbol{R}_b}{k' + 1}. \tag{1.6}$$

In this verification, geometrically irrelevant images are rejected, and relevant images are back-projected onto the query region in order to reject irrelevant visual words appearing in the retrieved images to formulate the better relevant next-round query by averaging the BoVW histograms of the original query and the verified images with selected relevant visual words. The architecture of AQE is show in figure 1.12. This sort of refining with selected relevant visual words achieves higher recall comparing with that of the first-round ranked list.

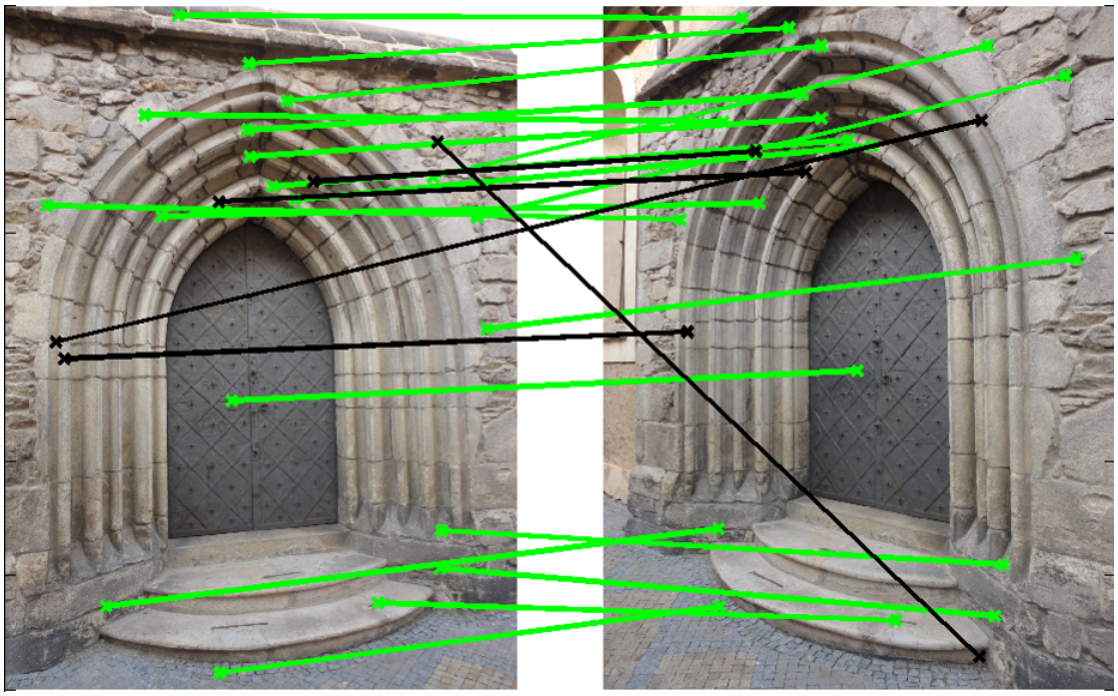


FIGURE 1.13: An example [a] of spatial verification between images by using RANSAC method. Green lines represent inliers between images. Black lines represent outliers.

[a]Image reference: https://cyber.felk.cvut.cz/theses/detail.phtml?id=360

A RANSAC like algorithm, e.g., LO-RANSAC that proposed by Lebeda et al. [33] , is typically used for the geometric consistency check as in figure 1.13. The *inlier threshold* parameter (th) is used to control the consistency, which thresholds the number of point pairs between two images (the query and one of the returned images) satisfying the estimated geometric transformation (scale, shift, affine, homography, etc.). Verified images ensure a higher quality second-round query $\boldsymbol{Q''}$. In obtaining the average (Eq. 1.5) for the next round query, the method takes the union of features of the original query combined with regions in returned images back-projected into the query region by the estimated transformation.

AQE imposes pairwise spatial constraints on the query image and each of the database images (see figure 1.14). The method lies at the heart of recent state-of-the-art visual QE methods. But in so doing it may narrow the range of the expanded query or miss information that is in the relevant images but not in the query (e.g., it can miss objects with occlusions or small objects with low granularity or noise). However, since AQE checks only the pairwise consistency between the query and each of the highly ranked images, its performance may be affected by slight degradations in the query image (see figure 1.15).

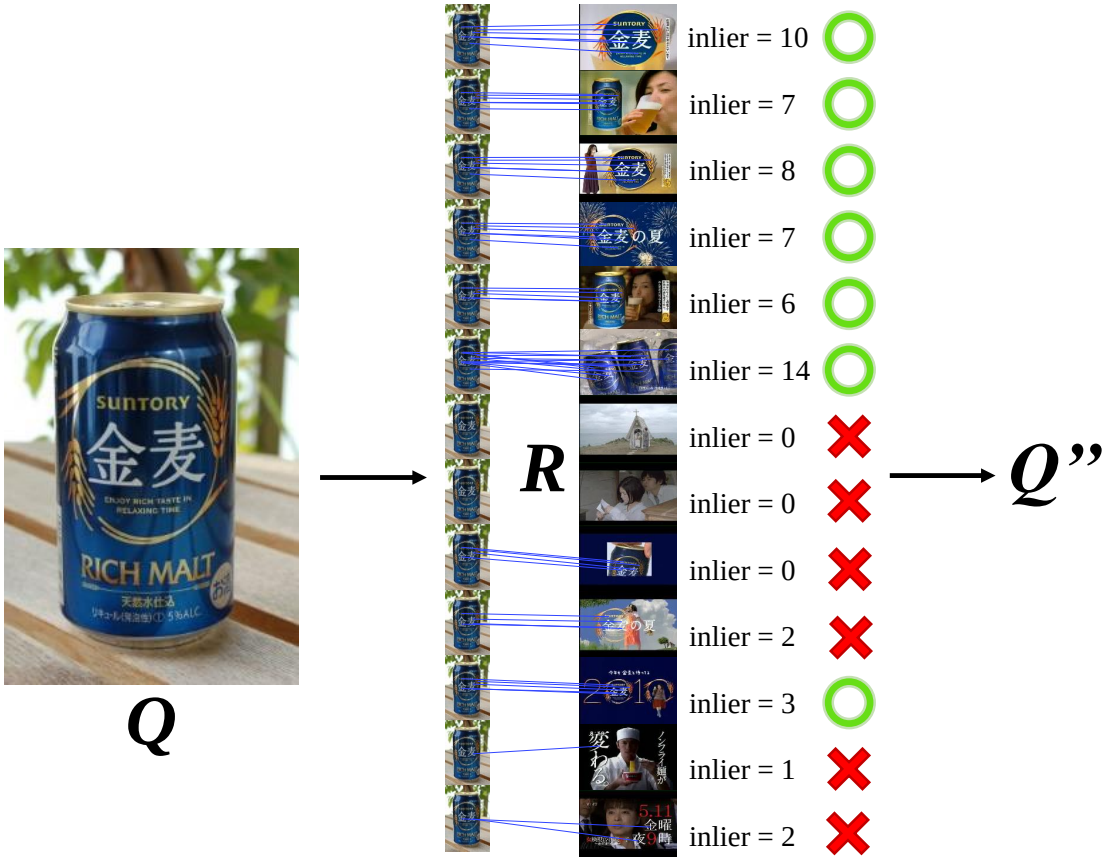


FIGURE 1.14: The figure shows a verification of a rank list $\boldsymbol{R}$ with a query image $\boldsymbol{Q}$. Inlier counts are calculated by using RANSAC methods. Only the images that pass a specific inlier threshold will be averaged together for making a second round query $\boldsymbol{Q''}$

1.3.4 Frequent Item Sets Mining (FIM)

FIM is aiming at finding regularities in transactions [34]. Well-known application of FIM is market basket analysis: given large number of lists of items bought by customers (a list is called a transaction), FIM finds multiple sets of items (called frequent patterns) bought together by many customers. Here, let us denote that $T = \{t_1, \ldots, t_{n_t}\}$ is a set of transactions, and each transaction $t = \{i_1, \ldots, i_m\}; i \in \mathbb{Z}$ is a set of items. A set of patterns is defined as $P = \{p_1, \ldots, p_{n_p}\}$, where each pattern p_i is a set of items. Each pattern has a corresponding *support*:

$$\begin{aligned} sup(p,T) =& \frac{|\{t \in T | p \subseteq t\}|}{|T|} \\ S =& sup(p,T) \\ P =& FIM(T,S) \end{aligned} \tag{1.7}$$

representing the fraction of transactions which contains the pattern p out of all transactions. The parameter *minsup* ensures FIM to output only patterns having larger

FIGURE 1.15: The example of problems when retrieving and an object by using mobile devices. Here are the example ordered from top to bottom and from left to right: image with a small object, low resolution image (after user pitched to zoom), noisy image (under the low lighting condition), blurry image (out of focused), inappropriate lighting conditions (reflection and back-lighting image), shaky image (query taken under motion), and occlusion.

Img. I_k	Trans. t_k
I_1	$t_1 = \{i_1, i_2, i_4, i_6\}$
I_2	$t_2 = \{i_2, i_5, i_8\}$
I_3	$t_3 = \{i_2, i_3, i_9\}$
I_4	$t_4 = \{i_1, i_2, i_4, i_7\}$
I_5	$t_5 = \{i_2, i_3, i_8\}$

Pattern	*support*
$\{i_2\}$	60%
$\{i_3\}$	40%
$\{i_8\}$	40%
$\{i_1, i_4\}$	40%
$\{i_3, i_8\}$	20%
$\{i_1, i_4, i_7\}$	20%
$\{i_2, i_3, i_9\}$	20%
$\{i_2, i_5, i_8\}$	20%
$\{i_1, i_2, i_4, i_6\}$	20%

TABLE 1.2: (left) Input simple transactions of top-five images. (right) Corresponding output patterns found with *minsup* of 10% (see section 3.2).

supports than a given value. When a pattern to be output contains a large number of items, FIM may generate an extremely large number of patterns because all elements of the power set of the pattern are by definition to be output. This problem is called the colossal pattern problem [35], and it may degrade the performance of FIM. Simple outputs of FIM are shown in table 1.2.

1.4 Problem summary

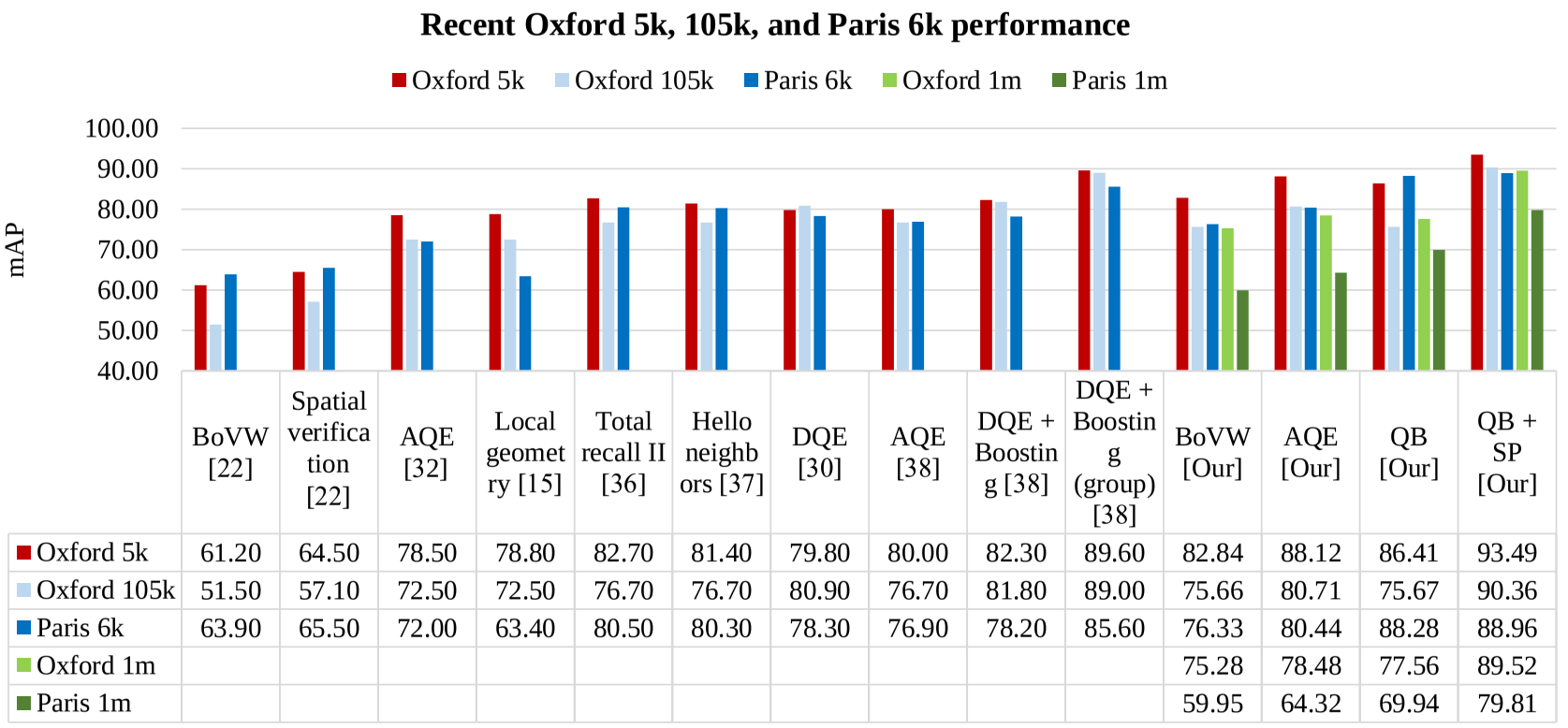
For BoVW, similarity score are calculated between images even if it contains only one match of the visual word, some retrieved images might not directly be related to the target object. This partially matched object or a small group of visual word does not always relevant.

AQE comes to solve such problem by utilizing spatial information. RANSAC like method has been employed for doing geometric verification on them. However, if a query image was taken in a good condition with highly visible of a target object, spatial verification will return quite good number of inliers. Since AQE strictly checks only the pairwise consistency between the query and each of the highly ranked images, its performance may be affected by slight degradations in the query image.

1.5 Contributions

We propose a new method called *Query Bootstrapping* (QB for short). The key idea of this variant of QE is to use the consistency among highly ranked images, instead of using only the pairwise consistency between the query and each of the ranked images. Doing so relaxes the over-dependency on a query that affects AQE, and thus QB may be more robust to the degradation and/or variation in the query images. We regard frequently co-occurring visual words in highly ranked images as relevant. We use frequent itemset mining (FIM) by Uno et al. [36] to efficiently find co-occurring visual words in highly ranked images, and we also use LO-RANSAC by Lebeda et al. [33] to check the geometric consistency of the highly ranked images and remove those that do not pass the check. FIM outputs frequent patterns, each of which is composed of a set of visual words, that co-occur frequently in the top-k highly ranked images. We then use the visual words appearing in the frequent patterns to formulate the next-round query by averaging together BoVW histograms of the original query and highly ranked (optionally geometrically verified) images with only the visual words in the frequent patterns. To do this, we propose *tf-fi-idf* as an extension of *tf-idf* that takes into account frequent patterns (*fi*) . This method requires the parameters to be carefully designed, namely, the *support* as the fraction of co-occurrences in the top-k highly ranked images, and the top k as the number of highly ranked images to be fed to QB. Example of second-round queries are illustrated in figure 3.2, with back projecting image segments of highly ranked images that contain visual words corresponding to frequent patterns into the original query space.

There have been a number of previous attempts at using FIM for image retrieval; however, very few of them have dealt with automatic optimization of such parameters. Here, we devised an adaptive *support* selection algorithm that returns both the minimum support *minsup* and maximum support *maxsup* in order to find the optimal faction of frequent patterns out of the top-k images. Moreover, we also created an algorithm that selects a suitable *inlier threshold* for the LO-RANSAC geometric consistency verification, which can be used to indirectly determine the value of k of the top-k highly ranked images. We tested our approach on standard Oxford and Paris benchmark datasets (Oxford 5k, Oxford 105k, and Paris 6k by Philbin et al. [22, 37]) and also with the extended version with MIR Flickr 1M dataset by Huiskes and Lew [38] (Oxford 1M and Paris 1M) and found that it outperforms a BoVW baseline, yields a significant performance improvement over AQE, and preserves higher robustness to query degradations. The overview of our performances comparing to the other baselines is present in figure 1.16.



	BoVW [22]	Spatial verification [22]	AQE [32]	Local geometry [15]	Total recall II [36]	Hello neighbors [37]	DQE [30]	AQE [38]	DQE + Boosting [38]	DQE + Boosting (group) [38]	BoVW [Our]	AQE [Our]	QB [Our]	QB + SP [Our]
Oxford 5k	61.20	64.50	78.50	78.80	82.70	81.40	79.80	80.00	82.30	89.60	82.84	88.12	86.41	93.49
Oxford 105k	51.50	57.10	72.50	72.50	76.70	76.70	80.90	76.70	81.80	89.00	75.66	80.71	75.67	90.36
Paris 6k	63.90	65.50	72.00	63.40	80.50	80.30	78.30	76.90	78.20	85.60	76.33	80.44	88.28	88.96
Oxford 1m											75.28	78.48	77.56	89.52
Paris 1m											59.95	64.32	69.94	79.81

FIGURE 1.16: The overview result of our methods (QB and QB + SP) comparing to the other methods, (from the left) BoVW by Philbin et al. [22], Spatial verification by Philbin et al. [22], AQE by Chum et al. [32], Local geometry by Michal Perďoch [15], Total Recall II by Chum et al. [39], Hello neighbors by Qin et al. [40], DQE by Arandjelovic [30], AQE by Chen et al. [41], DQE + Boosting by Chen et al. [41], and DQE + Group boosting by Chen et al. [41].

1.6 Outlines

This thesis is consisted with 7 main chapters and 2 additional appendixes. The organization of the chapters are mostly for for preparing knowledge for the readers from the fundamental foundations, applications, and the experiments as follow:

Chapter 1

We discuss several fundamental knowledge that required in prior to understand an image retrieval for grounding up basis of the reader. Starting from information retrieval theories explaining the way people search text documents. Following with a document indexing, similarity models, and term weighting scheme. The way to apply information retrieval to a visual problem and using SIFT feature to be described as a visual word. Hessian affine detector also a famous alternative to be used with SIFT framework. Clustering technique and the approximated methods is presented. BoW and BoVW are now almost the same, but AQE utilize better spatial information from images. RANSAC is used to verify images but not robust to query turbulences. We present a work based on an data mining technique. The result show our methods reach highest performance in this recent years comparing to the baselines.

Chapter 2

Literatures are being discussed here. Starting image retrieval works from the previous years with the smallest details of image content representation approaches, including feature extraction and the methods that applied this kind of visual feature on it, e.g., feature selection, feature matching, feature classification, and the libraries that we can use it easily. Following with the image retrieval system that was classified into a large scale system and a compact scale system, and the techniques to build up an inverted index database efficiently that suitable for SSD and HDD based storage devices. We also discuss about the cache management methods that help out system handle a very large scale database on a comparatively low spec of memory resource. Together with the frequent item sets mining tools that enable us to find co-occurrence object patterns and the recent approaches that adopt frequent item sets mining for visual problems.

Chapter 3

We present our propose method, namely, Query Bootstrapping (QB) in this chapter. Starting with the motivation of our work following with the work in details including

how can we process visual mining for the relevance list. We also present an algorithm for tunning *support* parameter adaptively. Finally, we introduce the way to integrate object mining result to the traditional BoVW framework.

Chapter 4

The extended version of our QB is present in this chapter. Beginning with the motivation and follow up by discussing the way to use spatial verification result appropriately. Also, another automated algorithm for tuning parameter is presented for finding an inlier threshold automatically on-the-fly.

Chapter 5

As the original FIM tool was not designed for a large number of item like visual words, we then show how to speedup mining process by using Galois connection on a full lattice. This technique is know as transaction transposition.

Chapter 6

We provide the overall evaluation of our method with the state-of-the-art in this chapter. Such as the impact of number of top-k relevance images, comparisons between fixed parameter and auto-parameters. Moreover, we evaluate the performance with various query turbulences for testing the robustness, e.g., object size problem, query resolution problem, noisy problem. Finally, the time consumption in details are also analyzed.

Chapter 7

Finally, the thesis is discuss the conclusion and the future work on this chapter.

Chapter 2

Literature review

"Life is like riding a bicycle. To keep your balance, you must keep moving."

— Albert Einstein

2.1 Existing image retrieval approaches

The goal of a large scale object-based image retrieval [15, 22, 30, 37, 42, 43] is to retrieve all images containing a specific object in a large scale image dataset, given a query image of that object. Generally, all the methods are required to provide good retrieval performance and be able to execute in a very fast or near real-time. Due to the restrictions of the technologies e.g., memory, disk space, transmission latency (network), etc. , we have to solve problems, at least to find a sweet point among limitations. Other challenges are large variations in the imaged object appearance due to changes in lighting conditions, scale and viewpoint, as well as partial occlusions.

2.1.1 Visual features

The question arises how to represent an image in a computer. We are not talking about pixel, which is too small to describe thing as image content. This representation has to be robust to large changes in imaging conditions, scale changes and differences in camera viewpoint. The standard approach is to extract many local image patches as an image

representation. This procedure relies on the assumption that two similar images will share a significant amount of local patches which can be matched against each other. Local patches are normally extracted by using detector, which is finding an interesting point called keypoint.

The major processes of image feature extraction are feature detection and feature description. The popular feature detectors include scale-space extrema detection on difference of Gaussian (DoG) [11, 29], affine region detections are [12–14], maximally stable external regions (MSER) [44]. And the alternative to the slow detector like SIFT are FAST [45, 46], Harris [47, 48], etc. , which are a magnitude faster than others as its name, but for different purposes alternatively for corner detection and edge detection.

In order to qualify if two images contain the same object is based on extracted image patches are matched. Normally, similarity computing between patch sets cannot be done by raw pixel colour or intensity. Since it is clearly not robust to changes in imaging conditions. Lowe [11, 29] then presented scale-invariant feature transform (SIFT) descriptors that are by design robust to such changes, as well as being discriminative enough to distinguish between different patches. From that point, visual related researches has been encouraged to work with local image features. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving [49].

The other image features inspired by SIFT are trying to improve the speed and several drawbacks. Gradient location and orientation histogram (GLOH) [14] reduces the dimension of descriptor to 64 instead of 128 on SIFT by using principal components analysis (PCA). Histogram of oriented gradients (HOG) [50] works very well as a descriptor for detect objects such as human when it was first introduced, as used in many computer vision researches that need to find human [51–53] and other objects. Speeded up robust features (SURF) [54] is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT. Since the speed of SIFT and even SURF is not quick enough for time mattered applications, researchers then introduce several binary features for real-time usages. Binary robust independent elementary features (BRIEF) (55) is introduced for a highly discriminative descriptor with relatively few bits and can be computed using simple intensity difference tests. Binary Robust invariant scalable keypoints (BRISK) [56] is presented with a key to speed lies in the application and computed in a scale-space FAST-based detector with combination of a bit-string descriptor. ORB is then follows for an efficient alternative to SIFT or SURF [57]. ORB is a very fast binary descriptor based on BRIEF, It claimed to has two orders of magnitude faster than SIFT, while performing as well in many

situations. However, it seemed in these days people are still love in SIFT performance, while prefer ORB in its speed on real-time application and a very compact code. At least, ORB is presented by a team who made an open source computer vision, namely, OpenCV [58].

Several alternatives based on SIFT are, ASIFT: a fully affine invariant image comparison method, Affine-SIFT [59] by pre-computing several affine transformed images using different camera angles then process all images with SIFT. And also with the improved of affine model within the feature detector, Perdoch Hessian Affine detector with SIFT descriptor is then introduced in [15, 16]. Most of visual features are implemented and can be easily integrated with many computer programming language and tools like, OpenCV [58], Matlab, Python, and the libraries like VLFeat [60, 61], Scikit-learn [62, 63], etc. .

Given sets of local descriptors extracted from all database images and the query image, large scale object retrieval duty is to efficient matching of the query descriptor set to the database descriptor sets and be able to output a rank of relevant images according to its similarity.

There are also several approaches that applied visual feature directly on their works. For example, query adaptive similarity by using feature to feature similarity by Qin et al. [64], an algorithm to discover useful features instead of confusing ones on a large scale dataset by Turcot and Lowe [65], relevance feature selection for a noisy medical image by Bugatti et al. [66], a fast kernel for feature matching for classification by Grauman and Darrell [67], an image classification for a large dataset by Deng et al. [68] that produce a relationship between images with its lexicon like WordNet by Miller [8]. And also a recognition works by Leung and Malik [19], Wallraven et al. [69], Nister and Stewenius [70].

2.1.2 Image retrieval systems

The full system of image retrieval have two trends, (1) a very large scale databases containing tens or even billions of images that performs with very high performance and huge amount of meta-data, thereby increasing the cost of computing power or (2) not so large system containing only small meta-data, some store only image id, also the descriptor level is very compact and back with some drawback about retrieval performance.

	Name	Total videos	Total images	Total visual features	Database size with spatial info. (in KB)
Small-scale database	Stanford-book	-	101	130,719	6,916
	Stanford-card	-	100	91,364	4,836
	Stanford-cd	-	100	99,939	5,360
	Stanford-dvd	-	100	135,372	7,280
	Stanford-landmark	-	501	628,337	33,452
	Stanford-paint	-	91	103,326	5,572
	Stanford-print	-	100	167,560	9,016
	Stanford-vdo	-	100	102,050	5,480
Large-scale database	Oxford 5k	-	5,063	17,390,270	928,764
	Oxford 105k	-	105,134	329,147,561	17,776,676
	Paris 6k	-	6,392	20,244,882	1,069,704
	Oxford 1M	-	1,005,063	798,393,536	43,239,632
	Paris 1M	-	1,006,392	801,248,148	43,381,504
	INS 2011	20,982	1,650,827	917,656,466	42,289,028
	INS 2012	74,958	2,256,930	1,908,832,917	91,422,180
	INS 2013-2014	471,526	7,837,877	9,814,391,541	443,438,120

TABLE 2.1: The dataset statistics show the details of compact scale database and large scale database in term of total number of videos (if available), total images, total number of visual features, and the database size (in kilobytes (KB)) including a spatial information. The datasets are, Stanford-MVS [71] for a small-scale, and for large scale are Oxford 5k,105k, and Paris from the work [22, 37], and even larger with adding MIR Flickr 1M dataset [38] into Oxford 5k and Paris 6k to build Oxford 1M and Paris 1M dataset.

2.1.2.1 Full system

A very large scale of image retrieval is aimed for executing the query search through a large scale image database in a sub second.

The important question to ask the is how scalable of BoVW based approaches? In our work we have successfully run our BoVW-based retrieval system on a single server which performed a search task using one CPU on an entire dataset of large scale images in less than a second. The statistic of building database is presented in table **??**.

As a framework is however firstly introduced in Video Google of Sivic and Zisserman [9], however the database is just a sampling of image one frame per second, thereby not so large. Real large scale is begin around the work of Philbin et al. [22], with a very large vocabulary size of 1M to 16M with a very high time complexity of k-means. Therefore, it is intractable to compute exact k-means for the suggested vocabulary sizes. Philbin et al. [22] shows that using an approximate k-means (AKM) to build clusters, namely, visual vocabulary, by using a randomized k-d tree is significantly fast with a complexity of $O(N_d \log(N_w))$, where N_d is the size of training descriptor and N_w is the size of visual words.

As TREC conference series of National Institute of Standards and Technology (NIST) published a very large video dataset [72] for the Instance Search task in the TREC Video Retrieval Evaluation (TRECVID) workshop. The work of Zhu and Satoh [73] proved the concept of using a very large vocabulary size of 1 million words from a standard BoVW framework proposed by Philbin et al. [22] together with a traditional SIFT feature [11, 29]. Some work [26, 37, 74, 75] proposed to work with soft assignment for visual word. However, we prefer to increase its descriptor discriminativity while reducing the effect of visual burstiness [76] from the quantization error, which can be founded as the high frequent word in figure 2.1. Then, with RootSIFT technique proposed by Arandjelovic [30], burstiness problem seemed to be helped since the shape of the cluster is in a kind of power law distribution (see figure 2.1). Also, with the affine region for feature detector from Michal Perďoch [15, 16]. All these ingredients are efficiently push Zhu and Satoh [73] performance on this large scale dataset to be beyond any others in TRECVID Instance Search 2011.

Later, a much larger dataset is included in Zhu et al. [77, 78]. The dataset archives are collected on three and a half years of full-day TV programs from seven Japanese TV channels. and five-channel (TBS, TV Tokyo, NET, FUJI and NTV) commercials into consideration. The dataset was prepared to be only commercial videos excluding news, dramas, etc. The video dataset is collected using recurrence hashing algorithm proposed in [79], was applied to detect commercials and clustered those detected commercial reruns into groups. Additionally, the work of Zhu and Satoh [73], Zhu et al. [77, 78] are kind of fully large scale system with a client-server based architecture (see figure 1.4).

Resource limitation

Although this performance is quite impressive, the resource requirements e.g., RAM and CPU are scale linearly with the number of features in the database. For every feature in the dataset, the inverted index database has to store the post list of an image id that contain such of visual id. For example, the image is encoded as a 64-bit integer (8 bytes). For a database of 1M images, assuming each image were assigned with 1k visual words. To build a basic inverted index, it takes about 8 GB of disk space. And to make a cache hit 100% for this inverted index, all this 8 GB of memory will be used by a search module (calculated with out data structure headers, usually 2-3 times of the actual data size is total size after included the variable headers (C++)). Thus, with a dataset 100 times larger, it is not possible to use the same BoVW architecture, as it is very expensive, but may possible with a single machine with 800 GB of RAM , or we need to change the single machine architecture to be able to execute the search module, in parallel using a distributed machines and distributed inverted index database [80] with hadoop and map-reduce style [81] , or even on cloud , etc.

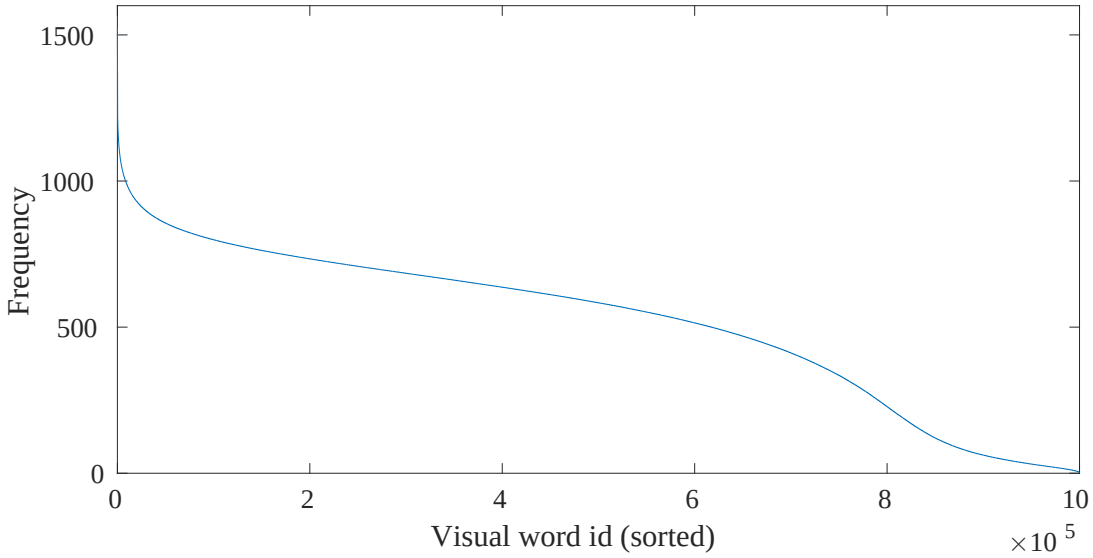


FIGURE 2.1: A cluster distribution of an Instance Search 2011 dataset shows the visual word clusters built by an approximate k-means algorithm (AKM) and quantized by an approximate nearest neighbor algorithm (ANN) has a shape seemly like a power law distribution.

The efficient ways to build a FAST-READ inverted index database

In order to solve the problem in case we have a very limited spec of a server, we build a framework as used in Kasamwattanarote et al. [82] with a very flexible in order to control the amount of memory usage during retrieval. As the cluster distribution as shown in a figure 2.1, we learn that, the high frequent words have more chance to be appeared in many images while the low frequent words are also important, but will not be visited that much. We then generate an inverted index with the following criteria for building a very FAST-READ inverted index for a general I/O:

- To save space on the memory, not all visual words need to be cached before hand.
- Inverted index is stored as a file on a general hard disk and it is unavoidable not to be read randomly, but it much prefers to be read sequentially.
- The high frequent visual words have higher probability that it will be matched with larger amount of images.

From the mentioned criteria, we then reformulate an index in-memory in advanced before flushing out into the disk. By sorting a lookup an index by its frequency, then write out all the post-list of documents according to its new assigned index. This process will make an inverted index readable sequentially from low to high identity (i.e., id 0 to 999999 in case of 1M vocabulary size), thereby the high frequent index will be grouped at the beginning of the database file.

We implemented an inverted index in two versions depends on an underlying I/O.

1. In case of storing and inverted with SSD , we can separated each visual word into each independent file as done in Zhu et al. [78]. For example, there will be 1 million files for handling 1M vocabularies. With SSD, we can perform a quick fetch operation within a very small latency, and also we can seek to any index with a constant time. Another benefit of storing inverted index independently is that, on each visual word index can be able to be updated on-the-fly in the case when we need to add more images or video frames. This usage seemed to be rarely use for an academic purpose, but on-the-fly updating the index will be necessary for the practical search engine e.g., Google.

2. In case of storing an inverted index with a normal HDD , randomly read is not recommended since accessing on a spinning disk require rather high latency. So, we store an inverted index into one BIG file, containing all visual words with its post-list image identification, which are also sorted by its *term frequency*. One reason is that, we need our retrieval module to fetch all cache from this slow HDD in sequential to reduce the hardware latency of moving a head and spinning disks. However, since we use 1 million words, the visual word histogram is usually sparse for each query search request. To handle this problem, we implemented an algorithm to group the cache request together for making more chance in accessing the cache mostly in sequential. The HUGE benefit of doing this is that, the next requests will get higher cache hit rather than cache miss, since the cache of around most frequent visual words will be cached faster and faster in each round of retrieval. The illustration of our caching scheme is present in a figure 2.2.

By performing a caching strategy, for each new query request, the cache hit rate will be richer in each round of retrieval. and also, it is hard and expensive to make a cache hit 100% when the initialize the retrieval system, especially this technique will save a lot of memory on a limited resource or a shared environment.

Moreover, we also implemented a cache management by using most frequently used (MFU) algorithm [83–85] as found in a visual memory management technique or a page replacement technique of the operating system in order to utilize a limited memory space under a machine that has limited resource. These memory management technique is applied in the work [28] that run with a very huge INS 2013 dataset with 443 GB of an inverted index database on a machine with 64 GB of RAM . This technique is also applied in and Kasamwattanarote et al. [82] as to prove the concept in handling a very large database and also very effective for our work, namely, PVSS: portable visual search service for researchers [86] that make a large scale commercial search [78]

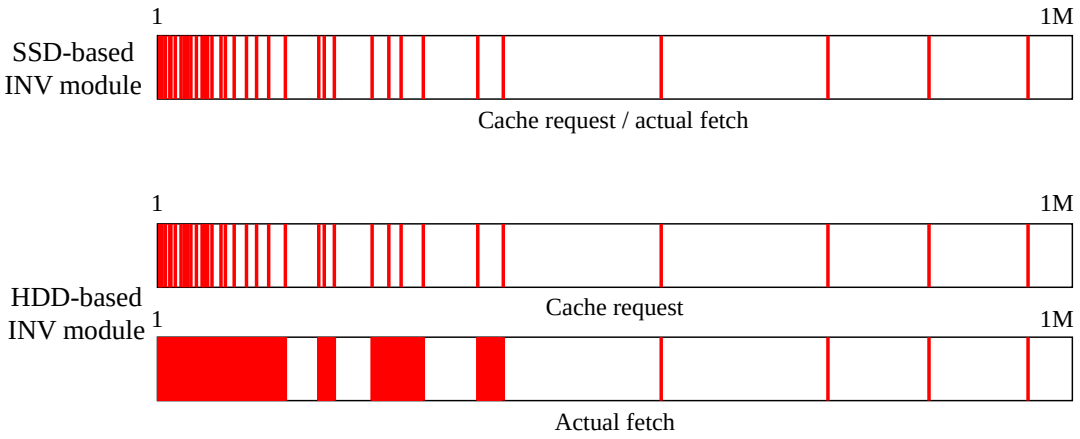


FIGURE 2.2: Our implemented cache fetching schemes for SSD-based inverted index database and HDD-based inverted index database of one-time query search. (top) For random access is possible for a fast I/O like SSD, the cache request and the actual fetch is at the same index. (bottom) For a slow I/O like HDD, the sequential access is preferred, so the cache request will be grouped together when the near requested indices can be detected.

possible (hosting a 52 GB of inverted index database on a limited 4 GB of virtual machine memory) to be served portably using a standard laptop PC with a server service hosting on the same machine by VM ware (see the retrieval interface in figure 2.3).

2.1.2.2 Compact system

Due to the size of a database and the limitation on computing resources, several approaches are also push effort for a compact system, major for a faster image retrieval and for adaptable to mobile device approaches. Firstly, Mikolajczyk et al. [14] proposes a SIFT like feature and reduce the descriptor from 128-dimension to 64-dimension by using PCA. Later, recently, binary features are proposed [45, 46, 55–57] for fast processing and very light weight of binary descriptor.

Thereby, these aforementioned features and its compression approaches provide a direct benefit to improve a speed and memory consumption for a large scale image retrieval systems and especially the methods will provide much usefulness for mobile approaches.

Most usage of mobile device in image retrieval is practically on a recognition problem. However, the problem is cast to an image retrieval system that need a specific spec for example, the system require a fast response to for an interactive user experience, the result must be reliable (comparing to a general image retrieval that returns many images as a rank list, but with a compact mobile device, only a top first image is important), a database size cannot be so large, a memory consumption should be very small, and the transmission to the server should not be that long.

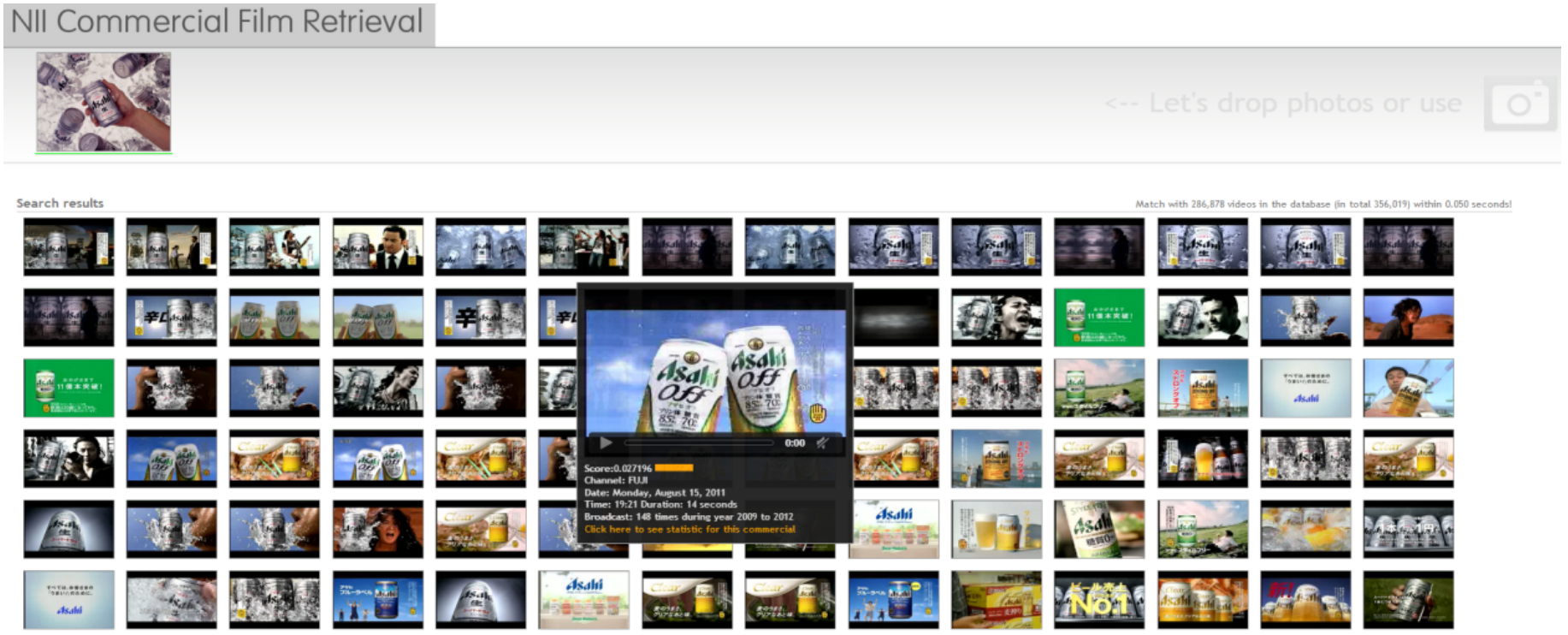


FIGURE 2.3: An interface shows the portable commercial retrieval system that run a large scale image database with our memory management techniques.

Girod et al. [87] presents a work that trying to model this problem as, a mobile visual search (MVS) systems that target a smaller dataset for some specific purpose e.g., recognize media covers, book spines, outdoor landmarks, artwork, and video frames in less than a second per query. Also, [88] also proposes a dataset for mobile visual search works, namely, Stanford Mobile Visual Search Dataset [71], which contains camera-phone images of products, CDs, books, outdoor landmarks, business cards, text documents, museum paintings and video clips. The dataset has several key characteristics: rigid objects, widely varying lighting conditions, perspective distortion, foreground and background clutter, realistic ground-truth reference data, and query data are also collected from various quality of mobile phone cameras.

Even though researcher also push effort on the other ways around to deal with a large scale problem. The work of Chen and Girod [89] proposes a way to compress a database and trim a lot of unnecessary visual words (e.g., from 1M to 250k) resulting to reduce the size of database and memory usages significantly, which is very efficient for running on a mobile device with less memory. The work from Chandrasekhar et al. [90] proposes a low-bitrate histogram of gradient (HOG) descriptor, and Ji et al. [91] also proposed an encoding algorithm for a low bitrate of signature.

The work of [92] proposes a way to reduce amount of data transmission significantly by mainly resize a query into smaller thumbnail. His work [92] reports the performance of retrieving with a small query image can be improved with a preprocessing on a server side (scaling up), and also with a compact feature descriptor computed from a client side that help explain a bit more for the specified query image.

And for a very compact work, Panda et al. [93] proposes a way to remove unnecessary information, compress a database, and so, to make a fully offline mobile retrieval system that can recognize and answer for a query request only a "label" of the corresponding object or landmark. This system need only precision @1 to be corrected.

The applications of a mobile visual search work that applied on a real-world problem are for example, sharing a mobile landmark in real-time by Dai et al. [94], product search He et al. [95], CamFind - a very deep back-end object recognition system by CloudSight [96] , and SATCH - a book cover search for a Japanese mobile company, KDDI [97] .

2.1.3 Feature bundling, packing, and embedding

Fisher kernel has been used with BoVW histogram sometime before. However, the work by Perronnin et al. [98], Jegou et al. [99], Jégou et al. [100] proposes ways to improve fisher kernel as by compressing the BoVW using Fisher kernel as called it as a

Fisher vector . The feature vector represented as a Fisher vector is significant smaller than BoVW, with using only few hundreds of bits per image, and significantly better in term of retrieval speed than any compressed BOVW approaches. Later, BoVW histogram compression is still being proposed as Sánchez and Perronnin [101] and [102] for a signature compression works.

Jégou et al. [103] proposes to an architecture alternative to BoVW framework, with a compact binary signature in additional to the inverted index posting list. This signature is called Hamming embedding , is used to the nearly matches of BoVW with the criteria on the Hamming distance between the query and matched signatures is small enough. The work of Jegou et al. [104] also proposes a fast memory-efficient method for ANN search adapt from their Hamming embedding (HE) . Jégou et al. [105] extract MiniBOFs by random aggregation of the binarized BoVW vectors. feature bundling [106] [107–109] propose a scalable method for discovering frequently co-occurring visual words by using MinHash [110]. And He et al. [95], Wang et al. [111] propose the use of hash bit for projecting a compact visual words code. All these approaches aimed to make a representative of image with more compact visual words for faster in search and lower memory usage.

2.1.4 Spatial information

We have been concentrating on image representation as local descriptors that quantized into BoVW histogram. However, image is not just only containing visual word as its contents. As the standard BoW on text retrieval has one key problem is that, the words list on a vector space model will lost its relationship or the orders of each word in a sentence. The problem also escalated into BoVW, as the visual word will lost its spatial location from the image space. To solve the problem, people try to integrate the spatial information into the representation or even on the final evaluation for ensuring that the final result will be relevance to the object spatial information too. The exam of our early idea on using spatial locations from top-k frequent visual words can be seen in the figure 2.4.

The 2D spatial location can be used in different step during retrieval.

1. *Early in the signature construction step.*
   The work of Wu et al. [112], Wang et al. [113], Zhang et al. [114] including a MinHash based by [115] focus on augmenting the local descriptor with a description of the distribution of visual words in the spatial neighborhood. Since the works are quite near duplicate search, are typically sensitive to feature drop-outs and scale

changes. Yao and Li [116] proposes a way to group visual features for recognizing human and object that has interaction. The work from spatial Philbin et al. [22], Wang et al. [117], Cao et al. [118] present the way to embedded spatial information into the signature using various approaches. And also including the work of spatial coding for web image search by Zhou et al. [119].

2. *During similarity calculation*
   Spatial information can be modified into the inverted index database. During the similarity calculation, the retrieved spatial information will be used for calculating the similarity as in the work of Zhang et al. [120], namely, geometric visual phrases (GVP) . collocation patterns [121]

3. *After the first round retrieval*
   This kind of using a spatial verification is at the object level. By verifying the query image to the retrieved images, an expensive spatial verification method like RANSAC will verify the set of keypoint location between them and return an inlier count as the output. This kind of technique is also applied on a spatial verification work for large scale retrieval, e.g., Michal Perďoch [15], Philbin et al. [22] and also in the work namely, weak geometric consistency (WGC) of [103]. The aim of the spatial verify the retrieved images is to re-rank the images order by the high inlier and move down the false matches to its end, thus improving (decreasing) the false positive rate.

2.1.5 Contextual information

Usually, BoVW representation will provide us the frequency of visual contents appeared on an image. Regarding to the target object, a query is generally be cropped for focusing the target object or even provided with an ROI as to control the background and foreground area.

By using this context information, calculating a similarity between a query image to each of an image in the database will be done regarding to a foreground area. Chum et al. [39] proposes a way to utilize this information by building a spatial context model that is trying to reject the score that come from a background area. And also Zhu et al. [122] proposes a work that utilize this context information for assigning a weight to background and foreground differently in dissimilarity calculation as the work namely, query adaptive asymmetrical dissimilarities.

Using multiple queries is also useful in order to utilize much wider of content descriptions with the different perspective. Multiple query images are used earlier in [123] as to cluster

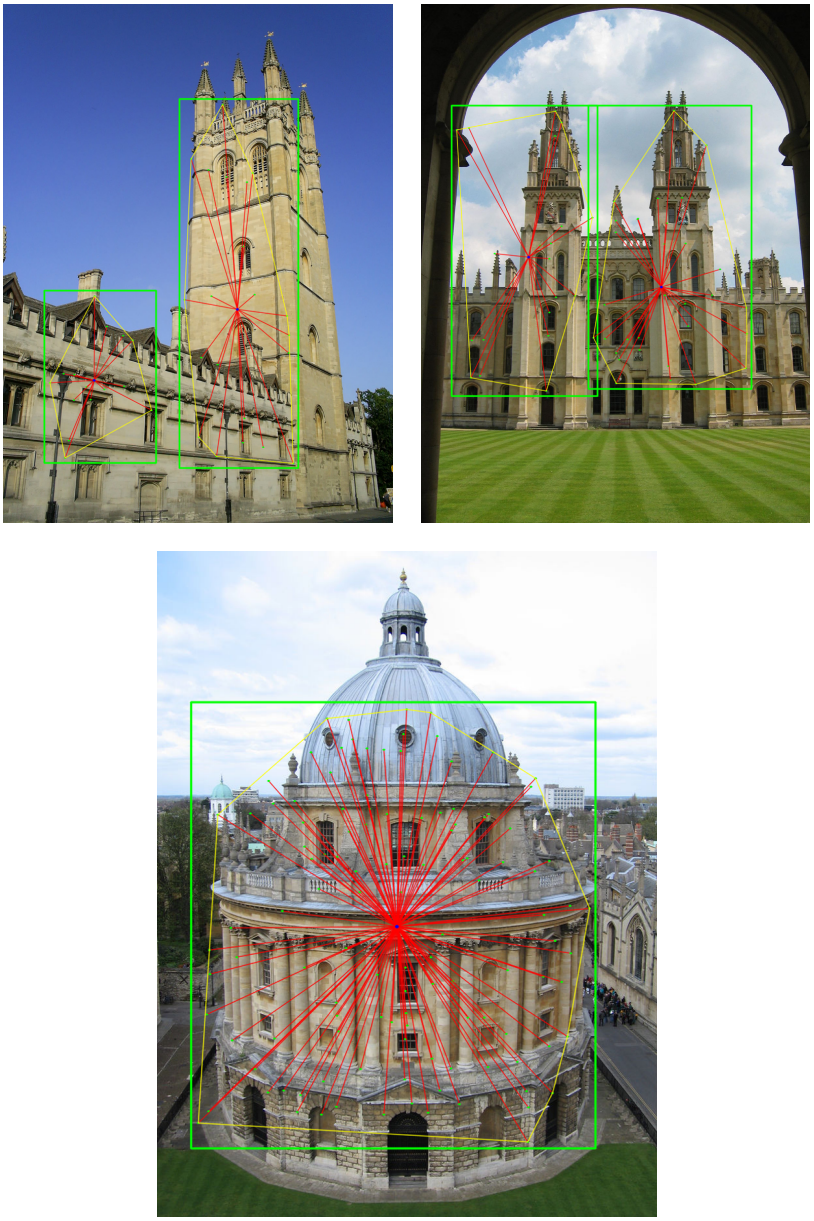


FIGURE 2.4: The figure shows the example of the objects found using our early idea to find the frequent objects by applying a clustering algorithm on the spatial location on all the frequent visual words that appear on the first round retrieved result.

multiple image results from the first round retrieved images for re-ranking. The work of web image search by Krapac et al. [124] is also utilize the related images on the web as multiple queries. And also, the works by Chen et al. [41], Arandjelovic and Zisserman [125] present the way to use multiple image queries and a group of query for a large scale system as well.

2.2 Approaches adopt frequent item sets mining (FIM) for visual problems

Association analysis is an approach for discovering interesting relationships between object hidden in large data set, which was initially applied to the market basket data for finding patterns existing within the sales of the products. This method help the shop to find the new opportunities in cross-selling like making a promotion of their products to the customers. Association analysis is first proposed as an association rule mining algorithm by Agrawal et al. [34]. Subsequently, Apriori algorithm and frequent itemset mining (FIM) is then proposed in Agrawal and Srikant [126], as a fast implementation of such algorithm, which is a widely used in pattern analysis at that time and also still very famous until now. Thereby, many researchers were attracted into this research field on mining association rules. Several approaches to improve this algorithms, e.g., Apriori-Hybrid and algorithm by Agrawal et al. [34], fuzzy association rule algorithm by Kuok et al. [127], FP-Growth algorithm by Han et al. [128] and the improved of its implementation by Borgelt [129, 130], linear time closed itemset miner (LCM) by Uno et al. [36, 131, 132], approximate frequent itemset mining by Yu et al. [133], Krimp for mining the compressed itemsets by Vreeken et al. [134] that is based on minimum description length (MDL) algorithm [135], etc.

Well-known FIM algorithms include FP-growth [128–130] , which achieves logarithmic computational complexity against the number of transactions but requires exponential memory complexity, and LCM [36, 131, 132] , which runs in linear computational complexity along with linear memory complexity. FIM tools are pervasively used for finding patterns in transactions of customers buying things from super market. In case of so many customer, the permutation of items through each basket can be quite pervasive, and can be too huge to be tackle by a limited set of computing resources. The example of the permutation of a very small items can be seen in figure 2.5.

However, we do not have to find all item sets, as the pattern can be overlapped and the core patterns can be raised in the item appearance frequency. Therefore, we can try to extract only the core pattern by using the introduced parameter, namely, *minimum support*. This parameter then served as an important for FIM, as the *minsup* threshold, for ensuring only the item sets that have a relatively fraction larger than the minimum of transactions are generated. By doing this, the number of patterns can be limited, to be able to control the quality of the output of FIM algorithm. In addition, [136] proposed an optional maximum constraint as calling as *maxsup* as for limiting the upper bound for the generated patterns to be discovered with maximum fraction of transactions. By setting both *minsup* and *maxsup* together, the pattern will be more focus on only a

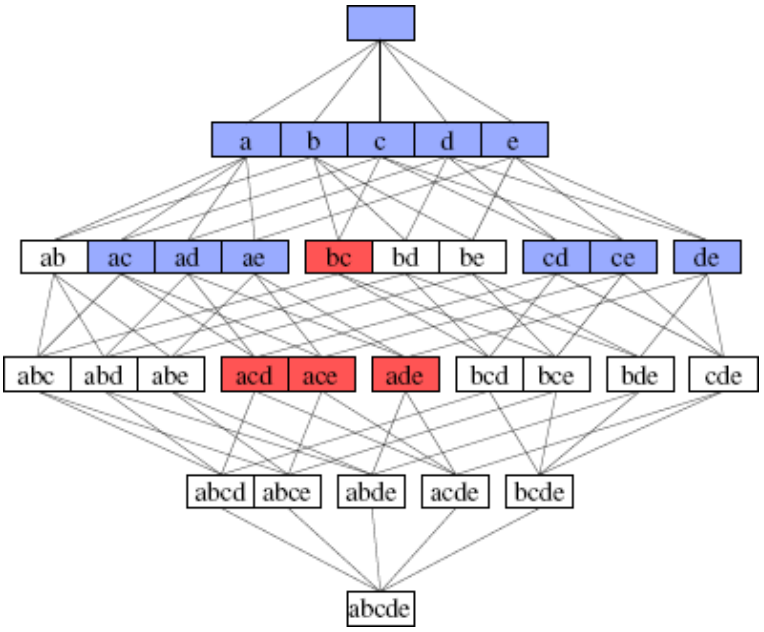


FIGURE 2.5: The example of the item sets permutation as a hasse diagram shows the permutation of the item a,b,c,d,e can be large as $2^n - 1$ as n is the total number of items. And the patterns can be much larger as increasing n. Taken from Borgelt [137]

specific patterns. Or even to control the output for judging multiple items with different criteria, the *minsup* and *maxsup* can be set with different ranges for multiple focuses of the generated patterns.

FIM has begun to be applied to the problems involving image feature earlier just a bit after BoVW has been introduced. The work of Quack et al. [138] proposes a way to construct a group of features by using a mining technique to detect an object that frequent and distinct. Nowozin et al. [139] proposes a graph mining for sub-structuring graph as a tool of image analysis. Action recognition is also being integrated with data mining technique on a its dense-feature as proposed by [140]. And also, the efficient way to used FIM on image classification problem, by Fernando et al. [141].

In another expect, FIM can also be integrated on multiple images as to utilize the context of other perspectives. With the use of multiple images, FIM can provides the meaningful co-occurring visual word patterns as object co-occurrences among multiple images or even video frames. In the context of multiple images or videos, and FIM has been used in video mining [142], visual phrase mining [121], mining of multiple queries [143], and mining for re-ranking and classification [144]. FIM is also reported to be able to find unambiguous spatially visual meanings [121]. However, FIM requires fine tuning of a sensitive parameter, namely, *minsup*. Most of the aforementioned methods heuristically assign a fixed *minsup* value or predefined number of patterns depending on the dataset. In contrast, we present an automated algorithm for tuning *minsup* and *maxsup* values on-the-fly in a way that depends on each individual query (see section **??**).

2.3 Evaluation procedure

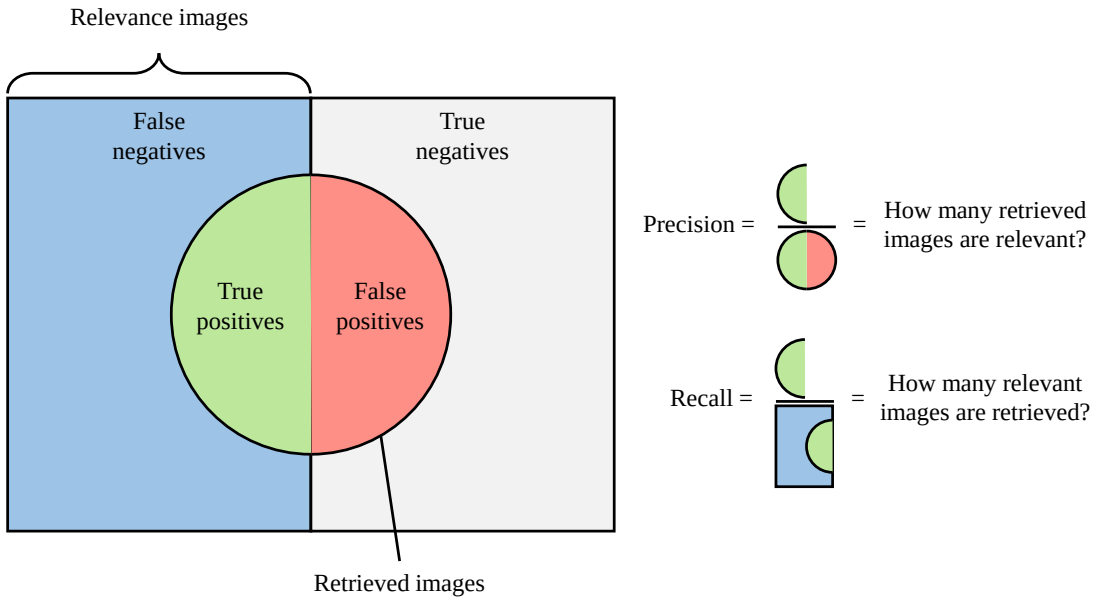


FIGURE 2.6: The precision and recall demo graphic of an image retrieval problem.

Performance evaluation through out this work is measure in term of retrieval quality for each single query by using a precision term of the precision-recall. An illustration of precision-recall is shown in figure 2.6. Precision is defined as the proportion of true positives in the retrieved images; recall is the ratio of retrieved true positives to the total number of positives for the query. In simple terms, precision measures the purity of the retrieved list, while recall measures what fraction of the total number of known positives is discovered.

Certain applications, like Google Goggles or Mobile visual search, which try to answer the question as only a recognition problem, are only interested in maximizing precision for obtaining a single correct result. For such applications, a useful performance measure for a single query is at precision@1, means the top first one is needed to be perfected. Large recall is also commonly required, for example 3D reconstruction requires a large number of relevant images in order to build an accurate 3D model. The corresponding measure for a collection of queries is the mean average precision (mAP).

Chapter 3

Query Bootstrapping: A Visual Mining based Query Expansion

"I have not failed. I've just found 10,000 ways that won't work."

— Thomas A. Edison

BoVW framework [9] is inspired from the tradition BoW work [6] and thus, there are many reasons that leads to the retrieval fail such as some keyword is missing. The example is that, a given query is provided with slightly different keyword, or a query contains too noisy of unimportant words (background clutter). Thus, the retrieval result will be directly affected from this. The same problem of BoW can be merely translated to BoVW, for example, the image query is provided with some object occlusion, or different camera angle, so the quantized visual words will be imperfect, slighly shift to the other words, and the result will be somewhat different. To help this problem, Salton and Buckley [31] proposed a blind relevance feedback where top retrieved results will be regards as relevances, and then it will be feed back as a second round query. The algorithm is expecting that, the missing keywords can be found from the other results which will be used in fulfilling of the missing part for the next round search.

Adapt to the visual search task, Query expansion [32] is a form of blind relevance feedback that work with image. for building a richer model of the query object and reissue the improved query. However, the word orders are lost in representing each

FIGURE 3.1: The problem of AQE when verifying relevant images using a provided query. (A) A normal query image with the verified number of inlier. (B) A low resolution of query image leads to lower number of the inlier, as in red, the relevant images has inlier count lower than the threshold (i.e., an inlier threshold is 4).

sentence during a BoW construction using a vector space model. Thereby, the problem of Bow is also inherited to BoVW as well. Chum et al. [32] presents a way to utilize the spatial information from the images. This work is named as Total Recall that try to improve the recall based on relevance feedback technique presented in [31] by using the average *tf-idf* vector of the query and the retrieved images. Additionally, the retrieved images are spatially verified with the query image by LO-RANSAC [33] in priori to perform the second round query. This technique is then becoming a standard of average query expansion (AQE).

AQE may help improves a recall as the technique can discovers some missing but informative keywords from the relevant images. However, in the case of a query itself is not quite clear describing on the target object, either by the camera angle is slightly shift or even the target object got occluded on its most part, constraining a verification by using this query will give much lower of an inlier count that yields too many false negative of such verification as shown in figure 3.1.

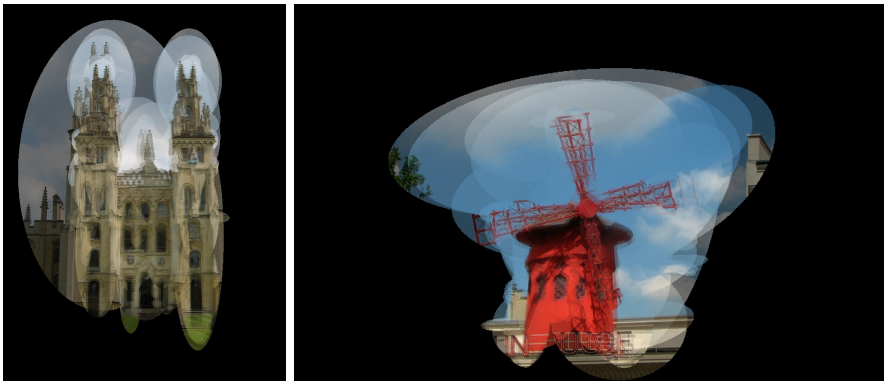


FIGURE 3.2: The illustration of a second-round query representing an object mined from multiple relevant images by using our method.

3.1 Motivation

AQE may fails with either an insufficient query quality or the retrieved images itself. The actual cause might be from one or both sides, but the result is the same as a verification fail and losing of tentative inlier counts (see Fig. 3.13b). Therefore, our aim is to relax this verification constraint by using the other ways around. The key idea is to use the object consistency among the relevant images itself (see figure 3.3) as to reduce the strong dependency from the query image.

By this reason, a data mining tool namely frequent itemset mining (FIM), which originally aims at finding regularities pattern in the shopping behavior of the customers of supermarkets, is a good candidate tool in finding regularities pattern within our the relevant images.

FIGURE 3.3: The idea is to use the object consistency among the relevant images itself as to relax the strong dependency from the query image.

3.2 Propose approach

We propose an integration of data mining technique for visual based query expansion, namely, *Query Bootstrapping* (QB for short). The key idea is to use the consistency among highly ranked images, instead of using only the pairwise consistency between the query and each of the ranked images.

We regard frequently co-occurring visual words in highly ranked images as relevant. We use frequent itemset mining (FIM) by Uno et al. [36, 131, 132] to efficiently find co-occurring visual words in highly ranked images. FIM outputs frequent patterns, each of which is composed of a set of visual words, that co-occur frequently in the top-k highly ranked images, and reported to be unambiguous spatially visual meanings [121]. We then use the visual words appearing in the frequent patterns to formulate the next-round query by averaging together BoVW histograms of the original query and highly ranked (optionally geometrically verified) images with only the visual words appeared in the frequent patterns. Example of second-round queries are illustrated in figure 3.2, with back projecting image segments of highly ranked images that contain visual words corresponding to frequent patterns into the original query space.

3.2.1 Design

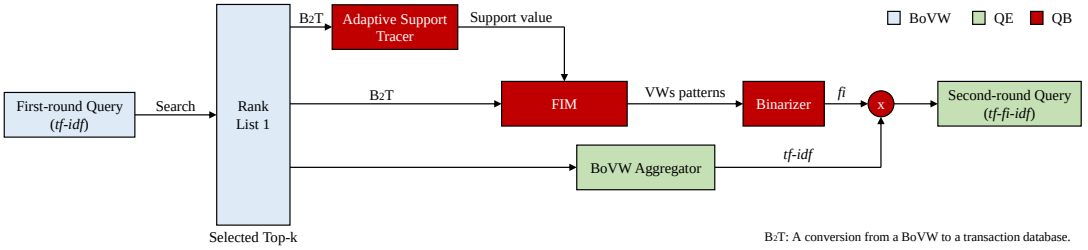


FIGURE 3.4: A framework overview of our Query Bootstrapping (QB) (scaled-up version see figure 3.5).

The architecture of our QB is designed as in the figure 3.4, by adding the mining components (in red) to a general query expansion (QE) framework (see figure 3.4). The mining components of our QB framework are frequent itemset mining (FIM) (section 3.2.2), adaptive support tracer (ASUP) (section 3.3), and the bag-of-visual-word integration (section 3.4). To handle the retrieved images by using FIM, we employed such mining components to process images separately from a general QE components. Then, QB will process and output the frequently co-occurred visual words (*fi*) as a result. Finally, QE and QB will merge the result for making a representative second round query. Note that, this framework has no process of spatial verification, which means to fully disconnect the highly query dependency as exists in AQE.

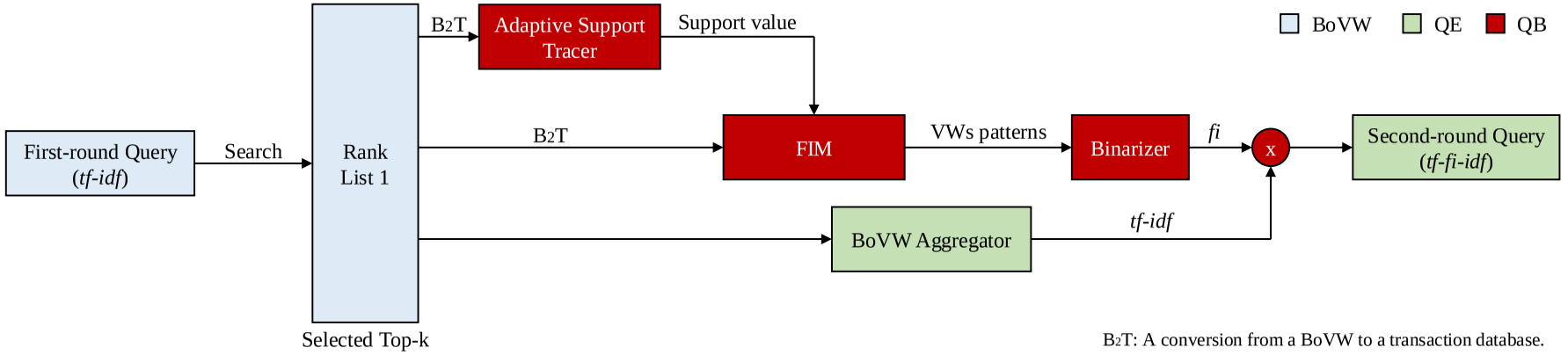


FIGURE 3.5: A framework overview of our Query Bootstrapping (QB) (Big).

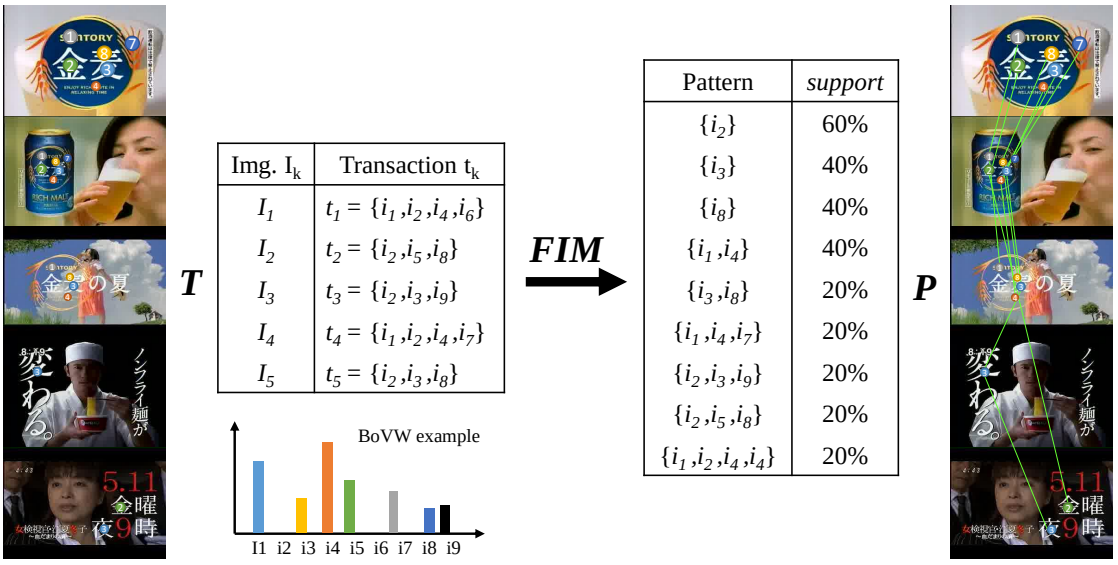


FIGURE 3.6: The illustration of the conversion from images that represented by BoVW to transactions of visual words collection, which are used as the inputs for a frequent itemset mining tool. The output patterns can be seen as co-occurrence of visual words as the object level configurations.

3.2.2 Method

In order to apply FIM to a ranked list of images, considering as a general FIM tool need to process data in a form of transactions and items as a market basket data. Firstly, we need to convert the list of images into a set of transactions. In this experiment, we regard only top 25 images to be used in FIM process. Therefore, each image in the list is regarded as a transaction, the quantized visual words as its items, and the set of images in the list as a set of transactions as known as a transaction database. The process of converting images to transactions, namely, we call this as a BoVW to transaction or B2T procedure, (see figure 3.6(left)). Given such a set of transactions, FIM outputs frequent patterns (see figure 3.6(right)) corresponding to frequently co-occurring visual words. As in the session 1.3.4, to process the transactions for getting patterns, the parameter namely, *minimum support*, is necessary to be specified. For example, the default parameter for FIM is usually $minsup = 10\%$, which will produce all patterns that have support more than or equal to 10%. The example of tunning a minimum support parameter and its corresponding output can be seen from a toy example output in figure 3.7.

3.2.3 Evaluation

According to the FIM process, we need to set the support parameter for each retrieval, that is, it is impossible to set the support parameter separately on each query topic.

Pattern	*support*
$\{i_2\}$	60%
$\{i_3\}$	40%
$\{i_8\}$	40%
$\{i_1, i_4\}$	40%
$\{i_3, i_8\}$	20%
$\{i_1, i_4, i_7\}$	20%
$\{i_2, i_3, i_9\}$	20%
$\{i_2, i_5, i_8\}$	20%
$\{i_1, i_2, i_4, i_4\}$	20%

(a) *minsup* = 10%

Pattern	*support*
$\{i_2\}$	60%
$\{i_3\}$	40%
$\{i_8\}$	40%
$\{i_1, i_4\}$	40%

(b) *minsup* = 35%

Pattern	*support*
$\{i_2\}$	60%

(c) *minsup* = 60%

FIGURE 3.7: The pattern outputs example corresponding to the specified *minimum support*, as to guarantee the minimum fraction of transactions to be covered. The total pattern calculated from a toy example from a figure 3.6. (a) setting *minsup* to 10% will produce all 9 patterns. (b) setting *minsup* to 35% will produce 4 patterns. (c) setting *minsup* to 60% will produce only 1 patterns.

Therefore, in this early experiment, we fixed the same support value equally on each query topic. Also, we evaluate it with different *minimum support* value from 5 to 95 and then run it in a batch mode for the following evaluation seen in figure 3.9.

The evaluation result of our QB in a figure 3.9 shown that each query has its top performance differently upon the designated *minimum support*. And the mAP for this evaluation is reported in figure 3.8. Due to the patterns output that belonging to each of relevant lists are not the same, regarding to its co-appearances of the visual object on each images, therefore, setting a global fixed support seemed not to be a good solution for our situation.

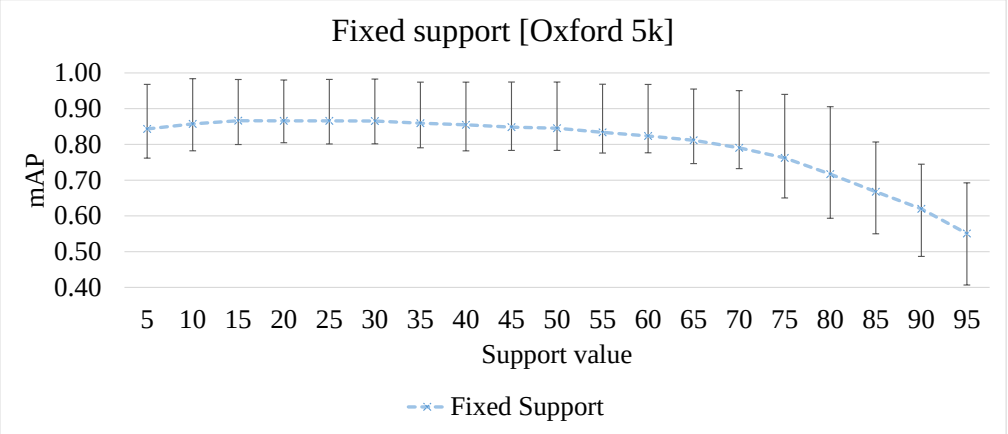


FIGURE 3.8: The figure shows the global mAP for each fixed *minimum support* value on an Oxford 5k dataset. The performance seemed to be at the highest when setting a fixed *minimum support* to 20%.

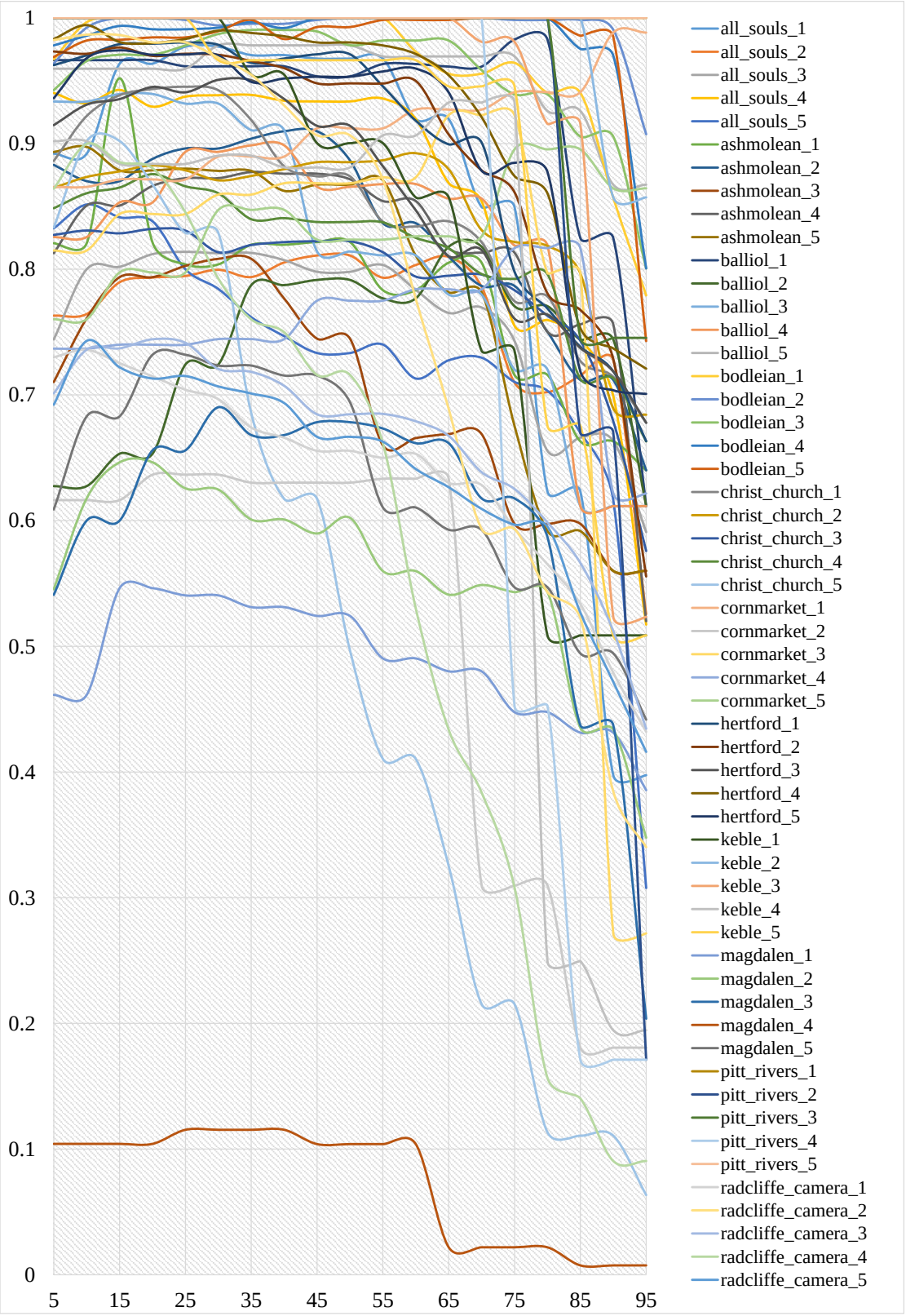


FIGURE 3.9: The performance evaluation in detail (locally) of various *minimum support* setting on an Oxford 5k dataset shows the best performance for each query can be achieved with different support value.

Pattern	*support*
$\{i_2\}$	60%
$\{i_3\}$	40%
$\{i_8\}$	40%
$\{i_1, i_4\}$	40%
$\{i_3, i_8\}$	20%
$\{i_1, i_4, i_7\}$	20%
$\{i_2, i_3, i_9\}$	20%
$\{i_2, i_5, i_8\}$	20%
$\{i_1, i_2, i_4, i_4\}$	20%

(a)
minsup = 10%
maxsup = 90%

Pattern	*support*
$\{i_2\}$	60%
$\{i_3\}$	40%
$\{i_8\}$	40%
$\{i_1, i_4\}$	40%

(b)
minsup = 30%
maxsup = 70%

Pattern	*support*
$\{i_3\}$	40%
$\{i_8\}$	40%
$\{i_1, i_4\}$	40%

(c)
minsup = 35%
maxsup = 45%

FIGURE 3.10: The pattern outputs example corresponding to the specified *minimum support* and *maximum support*, as to guarantee the minimum fraction of transactions to be covered and not larger than a maximum fraction of transaction to be covered. The total pattern calculated from a toy example from a figure 3.6. (a) setting *minsup* to 10% and *maxsup* to 90% will produce all 9 patterns. (b) setting *minsup* to 30% and *maxsup* to 70% will produce 4 patterns. (c) setting *minsup* to 35% and *maxsup* to 45% will produce 3 patterns excluding the pattern $\{i_2\}$, which has support higher than 60%.

3.3 Globally best with local optimized support parameter

FIM generates patterns under specific constraints on the *minimum support*. If we seek too high a support value, we may find patterns which co-occur very frequently in the top-k images, but may miss patterns (corresponding to target objects) which occur only moderately frequently. If we set too low a constraint for the support, FIM may generate too many patterns including background noise with the target objects.

The generation of patterns using *minimum support* is not enough as Lee et al. [136] suggest that the patterns can include mixtures of different patterns, which may correspond to target objects, such as buildings, as well as background objects, such as trees. Hence, their work present a way to use not only the *minimum support*, but also the *maximum support* to control the quality of pattern to be within a desired fraction of transactions as seen in figure 3.10, which shows that the patterns output can be controlled by both *minsup* and *maxsup* parameter.

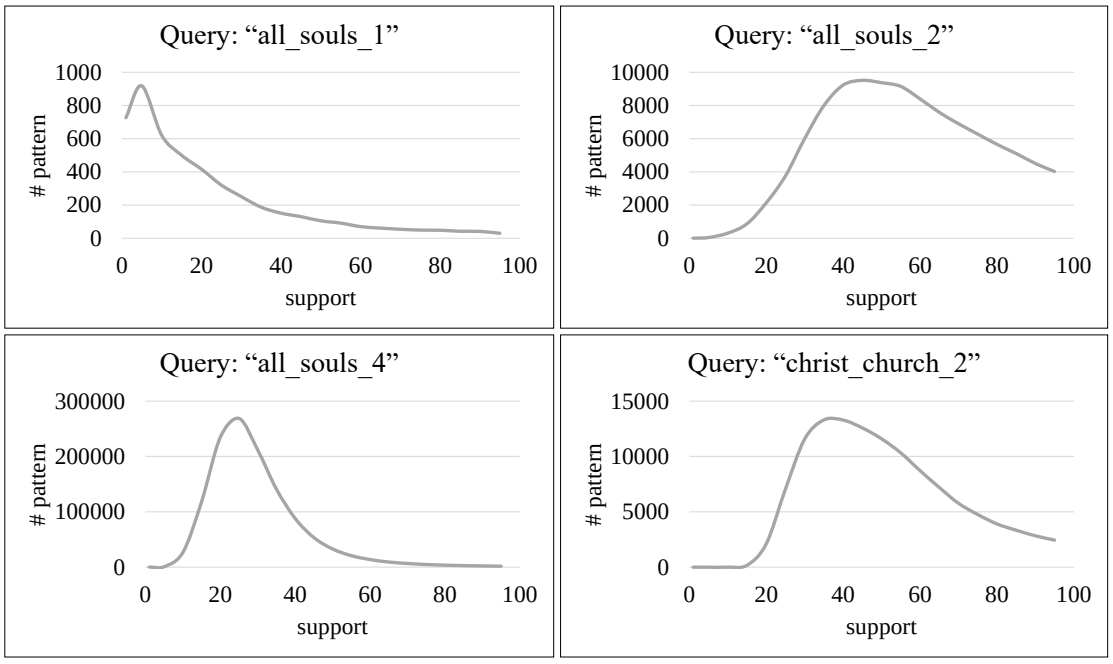


FIGURE 3.11: Plot of the number of patterns versus support reflecting the monotonicity property of the FIM principle for four different queries. The optimal *support* parameter is at the maximum number of patterns.

3.3.1 Motivation

Manually setting a *support* parameters perfectly is hard due to we cannot know the query image in advanced, so assigning a type of a query image for setting a support parameters is not possible. Therefore, we then learn from the experiences by running FIM on each query to check all the possible outputs providing with FIM tools, and we found one interesting value that has quite good relationship to the final mAP. This value is the number of pattern as the count of total of patterns after FIM has discovered. We conduct the experiment considering with intervals for support values. These intervals are the set of the range of $[minsup, maxsup]$ between 5-95% of support. Therefore, we assign these set as [5-10], [10-15], [15-20],...,[90-95]. then processes FIM according to the specified intervals. FIM will return the number of patterns together with the actual patterns We plot the number of pattern with the different support value ranges, as result, the figure 3.11 shows us a very interesting distribution according to the work in [136] that the patterns can be a mixture of different objects. However, our task is mainly focus on retrieving one object, as the result of our plots shown in the figure 3.11. As our observation on the plots, we make the assumption that a target object will alway shape the pattern into a unimodal distribution, for each unique query and the pattern around the dense area is what we need.

3.3.2 Method

In order to fine tune a *support* value for FIM according to our description in the previous section 3.3.1, we propose an adaptive support tuning algorithm, namely, ASUP, as to adaptively set the support parameter independently for each query. Accordingly, we can determine *minsup* and *maxsup* to be the support values corresponding to the maximum number of patterns, as shown in figure 3.11.

FIM restricts the output patterns to those having support values between *minsup* and *maxsup*. FIM takes two parameters, namely, *minsup* and *maxsup*, which restrict output patterns having support value between them (see figure 3.10). the pseudo code of our proposed adaptive support tuning algorithm (ASUP) algorithm is as follows:

Algorithm 1 Adaptive Support Tuning Algorithm

Require: $T \leftarrow B2T(\boldsymbol{R})$

1: **procedure** SUPPORT TRACER
2: $b\_sup \leftarrow 0, n\_sup \leftarrow 100$
3: *loop*: $b\_sup < n\_sup$
4: $s \leftarrow b\_sup$;(s as minsup)
5: $S \leftarrow s + 5$;(S as maxsup)
6: $P\_count[s] \leftarrow ||FIM(T, s, S)||$
7: $b\_sup \leftarrow b\_sup + 5$

1: **procedure** OPTIMAL SUPPORT SELECTION
2: $opt\_s \leftarrow P\_count_{max}.idx \times 5$
3: $opt\_S \leftarrow opt\_s + 5$
4: **return** opt_s, opt_S

In addition, FIM tools proposed by Uno et al. [36, 131, 132] (LCM) and Borgelt [129, 130] (FP-Growth), both have two operation modes. In one mode, it finds closed frequent itemsets which include all patterns greater than or equal to *minsup*. In the close itemset mode, the discover patterns can be rather large, even it is already small. On the other mode, it can finds maximal frequent itemsets, which are those of its immediate supersets are frequent. In the maximal itemset mode, the discover patterns can be very compact describing only the un overlapped pattern. Thus, the closed frequent itemsets mode tends to be slow especially when the number of patterns is large, while the maximal frequent itemsets mode runs very fast, and give only rough pattern.

To determine the optimal *minsup* and *maxsup*, we scan pairs of possible *minsup* and *maxsup* values with a fixed interval between them and inspect the number of patterns using FIM in maximal frequent itemsets mode. The *minsup* and *maxsup* pair yielding the maximum number of patterns are determined to be optimal. Finally FIM is then run in the close itemsets mode with the optimal *minsup* and *maxsup* to generate actual patterns. Note that using both *minsup* and *maxsup* help to improve the quality of

generated patterns on an exact frequent object hence contributes to the final retrieval performance, and faster since there are no need to discover unnecessary patterns outside this min/max support boundary.

3.3.3 Evaluation

Here, we compared the performance in terms of mAP based on QB + SP (chapter 4) on Oxford 5k for two different settings, one using a fixed *minsup* for all queries, and another using the adaptive support algorithm to adaptively select *minsup* and *maxsup* for each query. Figure 3.12 shows the results of this test. And as the result, *support* value had a strong impact on the final performance of Oxford 5k, and the best fixed *support* for Oxford 5k dataset was around 20-30 (blue line).

However, our adaptive support algorithm (ASUP) performed much better precision (red line) than the non-adaptive version (fixed support). Also, our adaptive support algorithm can serve much better variance (lower is better) among the different query, since a support parameter will be locally optimized for each query independently for performing the overall highest global performance.

The key successful of our ASUP algorithm is that, a support parameter will be locally optimized for each query yields a highest global performance as shows in figure 3.12.

This step assumes that given first round retrieval results are dominated by relevant images which contain target objects. Unfortunately, as Fig. 3.17 implies, this is not always the case. Otherwise FIM as well as parameter determination algorithm explained here may not work well. To cope with this problem, we can optionally employ spatial verification that we will explain in chapter 4.

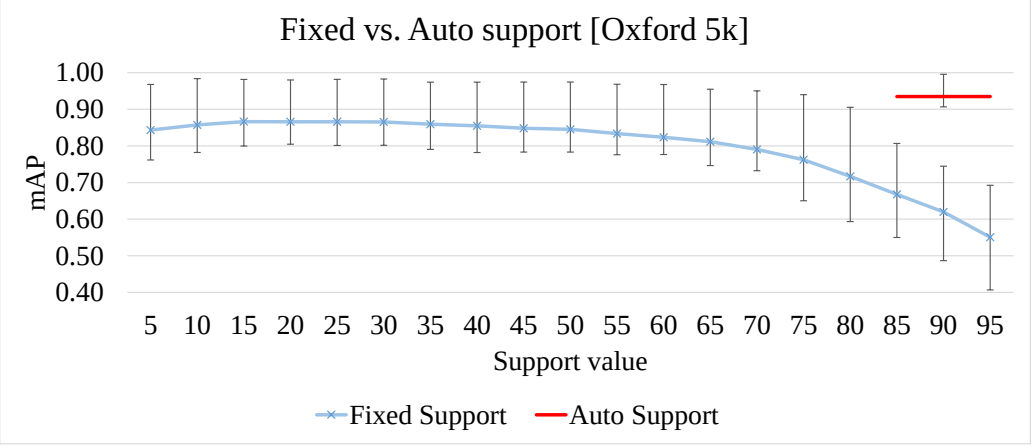


FIGURE 3.12: Retrieval performance of QB + SP on Oxford 5k dataset comparing the impact of a fixed *support* (blue line) and adaptive *support* (red line). The 1st and the 3rd quartile bar show that the adaptive method achieves lower variances among queries.

3.4 Integrating Query Bootstrapping to a BoVW

Up to now, we did used the result of our proposed QB and the adaptive support tuning algorithm as the frequent itemset guidance with the existing BoVW framework. However, in this session, we will explain how we applied this mining result to the standard BoVW framework.

FIM output result as the pattern of co-occurrence object among images, in order to incorporate the set of patterns obtained by FIM, we simply convert these visual word patterns into the weight that can be easily multiplied to the BoVW histogram. And in this early version, we simply binarize the pattern together as the visual word occurrences that directly come from a frequent itemsets (*fi*).

In order to utilize this *fi* weight, we extend the *tf-idf* weight for each quantized visual word by giving it the aforementioned binary term *fi* to the QE BoVW term as from the equation 1.5. *fi* is 1 if the corresponding visual word appears in any pattern in the set of patterns, and is 0 if the corresponding visual word does not appear in the set. The final weight, *tf-fi-idf* *tf-fi-idf*, is defined as follows:

$$\textit{tf-fi-idf} = \textit{fi} \times \textit{tf-idf} \tag{3.1}$$

Note that by setting all *fi* to 1, we obtain the standard query expansion.

3.5 Results

By integrating our proposed mining components on top of standard query expansion framework, our QB result shows the prominence performance as the overall mAP are reported in figure 3.12.

In this session, we select the examples of the result to show the advantage of using QB and additionally with adaptive support tuning algorithm (ASUP).

As our main objective of proposing QB is to relax the strong constraint over the query that may caused the spatial verification fail e.g., occlusion. So, the example case that the verification fail with the occlusion is shown in figure 3.13, while our QB utilize the object co-occurrences among relevant images, with the matching result shows the context keypoints were discovered through multiple images as shown in figure 3.13c.

(A) Query image

(B) Matching from AQE

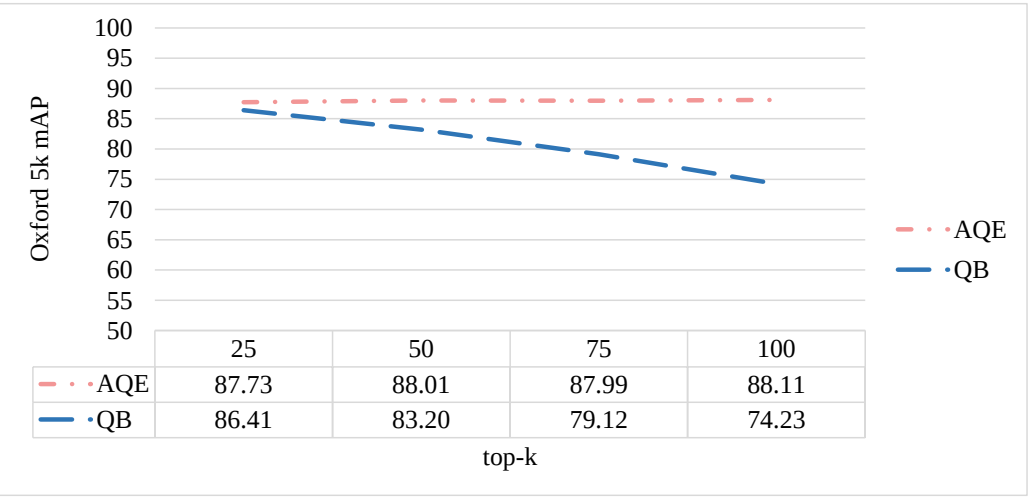
(C) Matching from QB

FIGURE 3.13: The example shows the matching points between query image and the reference image on the (a) A query image with its detected keypoints. (b) AQE lost several match keypoints due to the targeted object was partially occluded by a human. (c) QB got help by finding co-occurrence context outside of the target object.

For the burstiness problem, we show the comparison example between AQE and our QB in figure 3.15. As result, it shows that, by using *maxsup* we can remove those matches that highly appeared individually on each image, buy these match do not have much relationship to the target object. So, this results to better relevant matches with less burstiness problem.

And for the quantization error problem, we also show the comparison example between AQE and our QB in figure 3.16. As result, it shows that, some quantization error of the visual words are appeared on AQE matching results, how ever, our QB matching result shows much better relevant matched between images.

In some cases that the first round retrieved images were dominated by the irrelevant object, our QB will produce the result much worst than using AQE. For example in the figure 3.17, we show the retrieved images using standard BoVW (figure 3.17b) and our proposed QB method. As result, the within top 10 images of standard BoVW, there are only 7 true positive images were retrieved. However, in this experiment, we regard top 25 images as relevances and feed all these 25 images to QB. It means that, all the rest 18 images are irrelevance that dominated the result. Hence, FIM find the patterns respect to the other object (tree) rather than the target object (building), which finally lead to the negative impact of the second round result. We also conduct the experiments to show the more irrelevant images are include (top-k), the more noise in mining pattern using FIM, as reported in figure 3.14



	25	50	75	100
AQE	87.73	88.01	87.99	88.11
QB	86.41	83.20	79.12	74.23

FIGURE 3.14: The mAP comparison between AQW and our proposed QB shows the QB performance is decreasing when increasing the top-k relevant images.

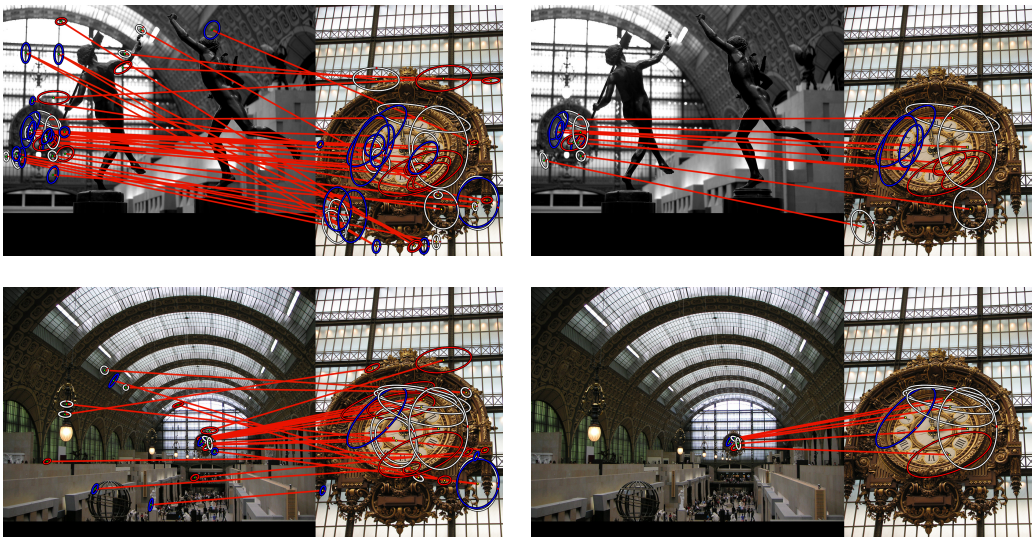


FIGURE 3.15: Burtiness matching between a database image and a query shows that our QB + SP (right column) gives better relevant words, while AQE (left column) yields irrelevant tentative matches in some cases.

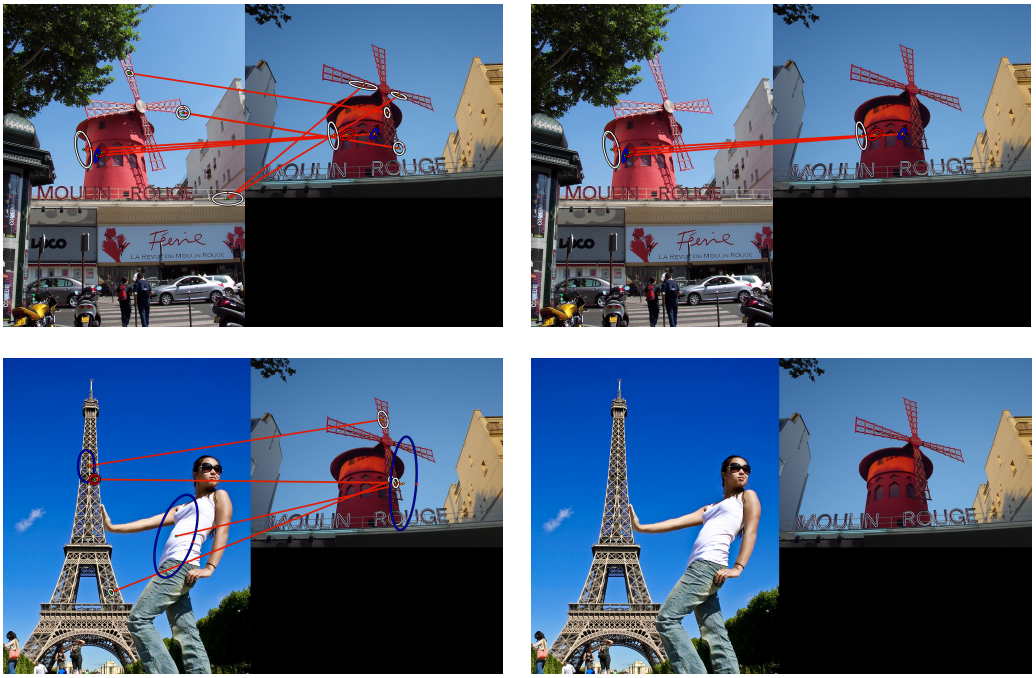


FIGURE 3.16: The matching between a database image and a query shows that our QB + SP (right column) gives better relevant words, while AQE (left column) yields irrelevant tentative matches in some cases.

(A) Query image

(C) Top 10 relevant images retrieved (AP = 100%) by BoVW and its top 100 true positive list in green.

(E) Top 10 relevant images retrieved by QB (AP = 9%) and its top 100 true positive list in green.

FIGURE 3.17: Example of QB failure when images retrieved in the first round were dominated by another object (a tree) rather than the target query (a building). Without a spatial verification for the query object, QB produces a result closer to the dominated one.

Chapter 4

Query Bootstrapping extended

"We have to continually be jumping off cliffs and developing our wings on the way down."
— Kurt Vonnegut

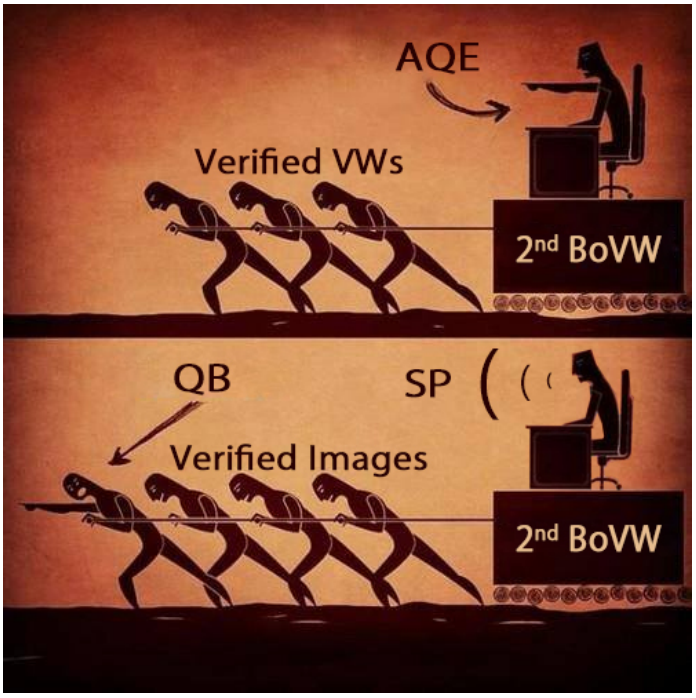


FIGURE 4.1: The image comparing AQE and QB + SP with the internet meme. (top) AQE as a boss to control the **verified VWs** for making a 2^{nd} query. (bottom) QB as a leader with hints from SP to make a 2^{nd} query using **verified images**.

We proposed a method, namely, Query Bootstrapping (QB) in chapter 3, as the variant of QE that is to use the consistency among highly ranked images, instead of using only the pairwise consistency between the query and each of the ranked images. As the objective is to relaxes the over-dependency over a various abnormally query situation. QB may be more robust to the degradation and/or variation in the query images.

4.1 Motivation

As the proposed QB as a visual mining for query expansion, we fully disregard the integration of spatial verification which is integrated in AQE and caused the aforementioned problem as described in section 3.5. We found the problem exists when the top-k rank got dominated on the retrieved images from the first round BoVW which may partially contains just part of object that visually look similar. This problem leads to the bad experience of our QB. In this context, FIM will find the most correspondence object out of the selected top-k images. And in case of such top-k list got dominated by irrelevant object the result of FIM will be totally different from our expect. There is a meme from the internet about the behaving like a leader is better than behaving like a boss; a boss is who commands the worker with only his own power, however, a leader is who lead their team with his guidances. Comparing to AQE vs. QB as in the figure 4.1, we shown that our QB is as a leader who listen to the boss (SP) to guide their team (verified images), for better in cooperation among all of them.

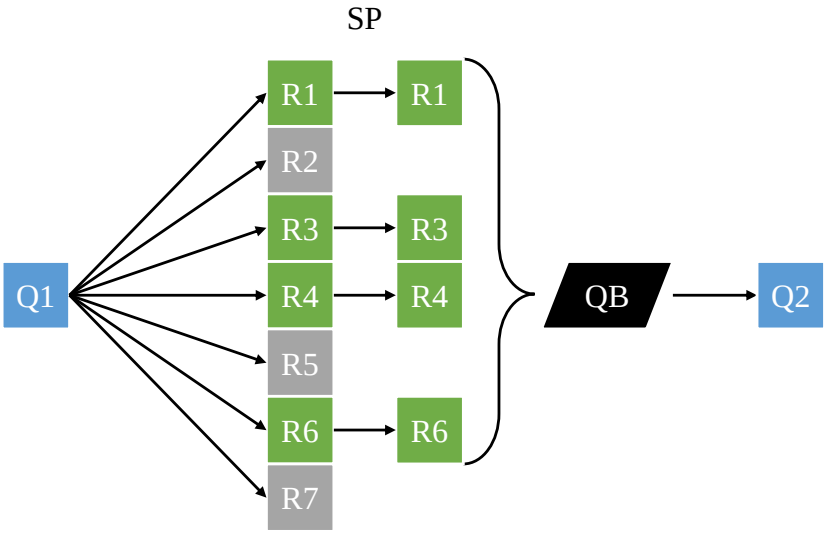


FIGURE 4.2: An illustration when QB takes an images verified by using SP module.

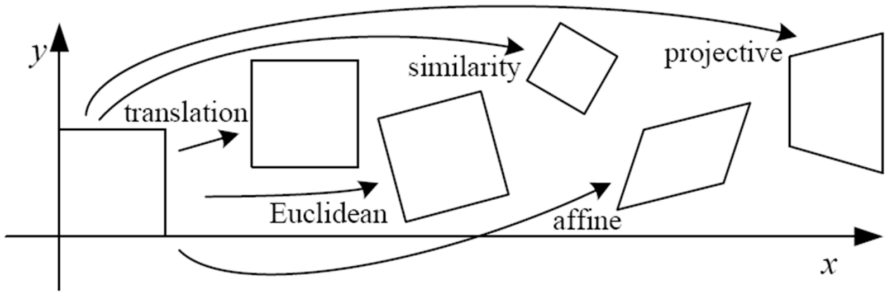


FIGURE 4.3: The projective transformation example shows the coordinate are transformed into different result according to the specified transformation parameters. The figure was taken from [145].

4.2 Propose approach

Therefore, in this chapter, we propose a way to bring back spatial verification method as to give hint on which target object should be focusing on during a mining step for QB. Our method shows the way to properly integrate spatial verification (SP) method as shown in the figure 4.2 that meant only the relevant images in term of visually similar of its object spatial information will be selected and be processed by our QB.

We regards the spatial coordinate (x,y) as key information for objects to be verified as relevant. By employing LO-RANSAC to calculate the homography matrix [1], which is a linear projective transformation relating two images, and verify if the translated coordinate is inlier between two images using a pinhole camera model . So, the homography matrix is defined as follows.

General homography matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.1}$$

Translation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.2}$$

Scaling:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.3}$$

[1] More details is summarized in the lecture of Fei-Fei [145]

Rotation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.4}$$

Shearing:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.5}$$

where x, y is the coordinate for one image, x', y' is the coordinate for another image, and $h_{11}...h_{33}$ are the parameter for transforming the coordinate, which will result to transform an image as specified degree of freedom, e.g., translate (eq. 4.2), rotate (eq. 4.4), (eq. scale 4.3), shear (eq. 4.5), affine, and perspective and the transformation result is shown in the figure 4.3. Hence, the any points between images that have no or less error (on the euclidean space) after being transformed using the calculated homography matrix are called as inliers.

4.2.1 Design

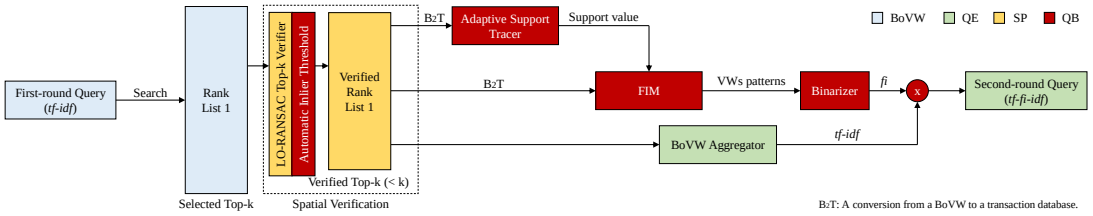


FIGURE 4.4: A framework overview of our Query Bootstrapping (QB) with an integration of spatial verification module using RANSAC (SP) (scaled-up version see figure 4.5).

In order to employ SP to our QB properly, we add the SP components (in yellow) in prior to our main mining components (in red) to filter out the irrelevant images in prior to the mining process, which result to a compatible to the designed architecture of our standard QB (in section 3.2). We called our extended QB architecture as QB + SP or easily as QBSP (QB combined with optional spatial verification), which is illustrated in the figure 4.4.

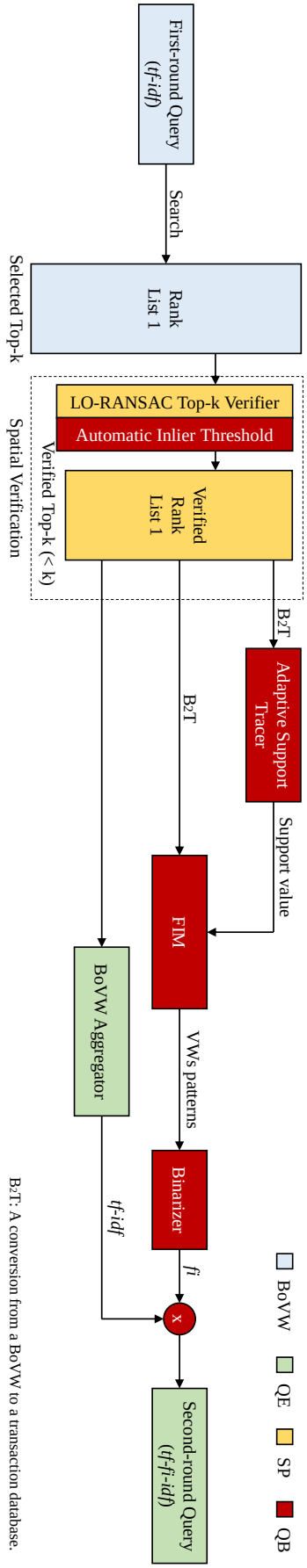


FIGURE 4.5: A framework overview of our Query Bootstrapping (QB) with an integration of spatial verification module using RANSAC (SP) (Big).

The overview process of our QB + SP framework works from left to right of the figure 4.4 as follows.

1. BoVW is extracted from the given query image and the first round result is retrieved with total of k images.
2. Spatial verification step is applied on top-k retrieved images and projected the k' verified images for the rest of the QB process.
3. QB processes the top-k' images for finding the correspondent patterns regarding to the adaptive support parameters.
4. QE aggregates all the top-k' images together for *tf-idf* for being ready for directly multiplied with the *fi* from the previous step.
5. The second round query is produced as the integration of query expansion + data mining + spatial verification as our *tf-fi-idf*.

4.2.2 Method

We use LO-RANSAC for a spatial verification process. LO-RANSAC generates a verified ranked list from a query image and the images in the retrieved list. LO-RANSAC actually finds the maximum number of point pairs between given two images that are geometrically consistent in the way relying on a homography matrix . Such point pairs are called inliers. In order to verify which image is an inlier, images having more inliers than a threshold are called verified images, while the rest will be counted as outlier images. This threshold is set as a criteria for filtering relevant images comparing to the inlier count calculated from such image.

In this step, we do not know how much inlier threshold we should set. Although, as general knowledge, the higher threshold will result to refine much higher quality of relevant images, there are the cases where we cannot find any survived relevant images that has an inlier count higher than the threshold. The reason might be because of the query image itself and/or together with the retrieved image, such as the view angle is different, the target object is small, occlusion etc. , that caused the low number of inlier count contrastingly to what we expected.

4.2.3 Evaluation

Therefore, our first experiment for this is to evaluate our QBSP comparing to AQE at each fixed inlier threshold. We set the inlier threshold into 5 values, e.g., 3, 5, 7, 9, and 11. We tested the experiments with the dataset Oxford 5k, 105k, and Paris 6k. So, there are 825 runs for each approach, and 1,650 run in total. The result is reported in the figure 4.6, which shows the performance result comparing to the different inlier threshold. By setting this inlier threshold too low can drop the performance for QBSP. On the other hand, setting this inlier threshold too high might not achieved the best performance on QBSP as well. For the AQE result, on the Oxford 5k dataset, setting this threshold higher caused a little drop of the retrieval performance. On the Oxford 105k dataset, setting this threshold higher will increase a little performance. And on the Paris 6k dataset, setting this threshold higher will drop the performance as well as QBSP did on this dataset. Therefore, in this section, we cannot conclude the best criteria for setting such threshold, since setting this threshold will not produce the same effect of retrieval result on different dataset.

However, generally, several existing approaches try to set this threshold manually according to their experiences, or until they found the result is good enough, then the threshold will be used in every verifications through all the datasets.

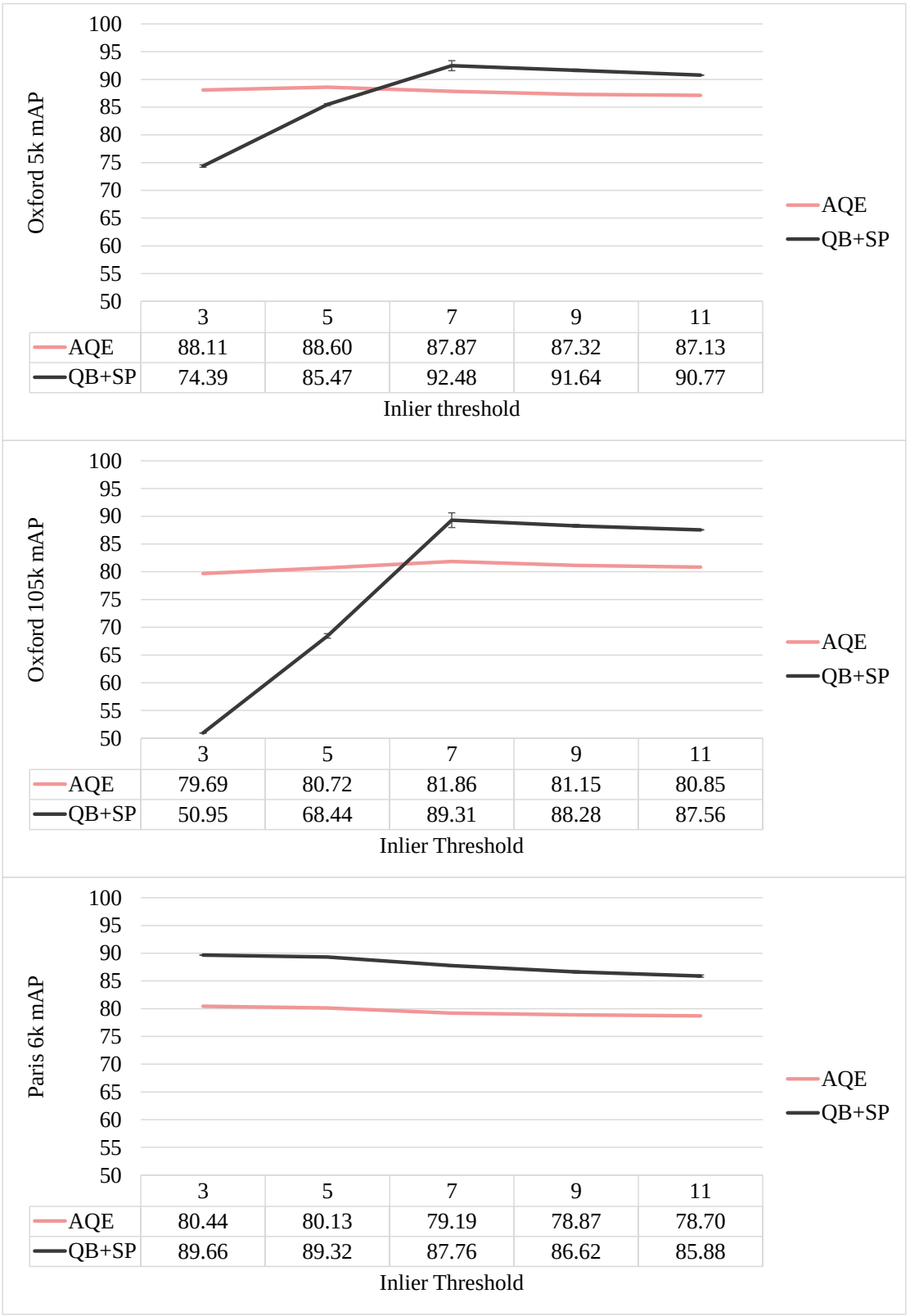


FIGURE 4.6: The evaluations of different inlier thresholds tested with AQE and QBSP. There are 5 thresholds for the experiments running on 3 standard datasets (Oxford building 5k, 105k, and Paris landmark 6k). The figure shows setting different inlier threshold can result to different retrieval performance. Setting the parameter too low can drop the performance for QBSP. Setting the parameter too high might not achieved the best performance on QBSP as well. Also, the criteria for of setting the threshold are not affect to the same result on different dataset.

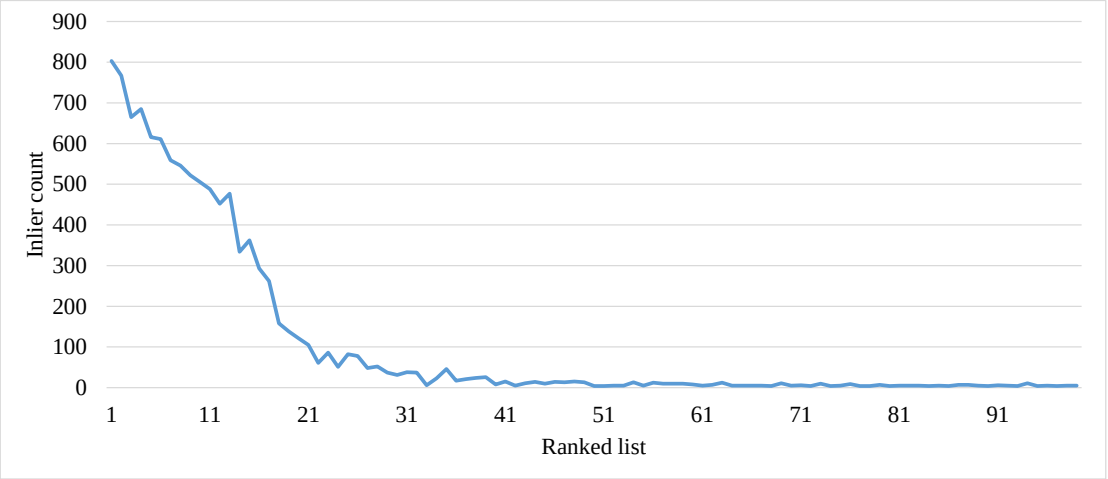


FIGURE 4.7: The plot of raw inlier counts that evaluated by using LO-RANSAC on a sample ranked list.

4.3 On-the-fly selecting inlier threshold

As we know, too tight of an inlier threshold may generate very few (or no) verified images, while too loose of an inlier threshold is equivalent to no spatial verification. Thereby, in this section, we propose an algorithm to adaptively tune this inlier threshold on-the-fly depends on how we see the result and how each ranked got the number of inlier count using LO-RANSAC. We named it as an Adaptive *Inlier Threshold* (ADINT) . According to the raw number of inlier we got from a ranked list as in figure 4.7,m which has a direct relationship to the rank of image. We assume that images having target objects tend to have large numbers of inliers, while images without target objects tend to have few inliners. So, from this observation, we target the new objective for clustering images into 2 groups, as an inlier image group, and an outlier image group, or aim to filter out irrelevant images by using its correspondent inlier count.

4.3.1 Motivation

Manually setting an inlier threshold for RANSAC-like method is done widely by several approaches. In case of easy and clear to recognize the object, an inlier count obtained from each of ranked list will be corresponded to visibility of an object that visually appeared in an image. Therefore, this imply the relevant images will contain quite high number of inlier count. In contrast, the lower ranked images are mostly reflect the low amount of inlier count. By this reason, setting an inlier threshold for filtering relevant image should not be fixed at some certain value, otherwise, the relevant images might be regarded as irrelevant if it cannot provide such high number of inlier comparing to the reference image like a degraded query image.

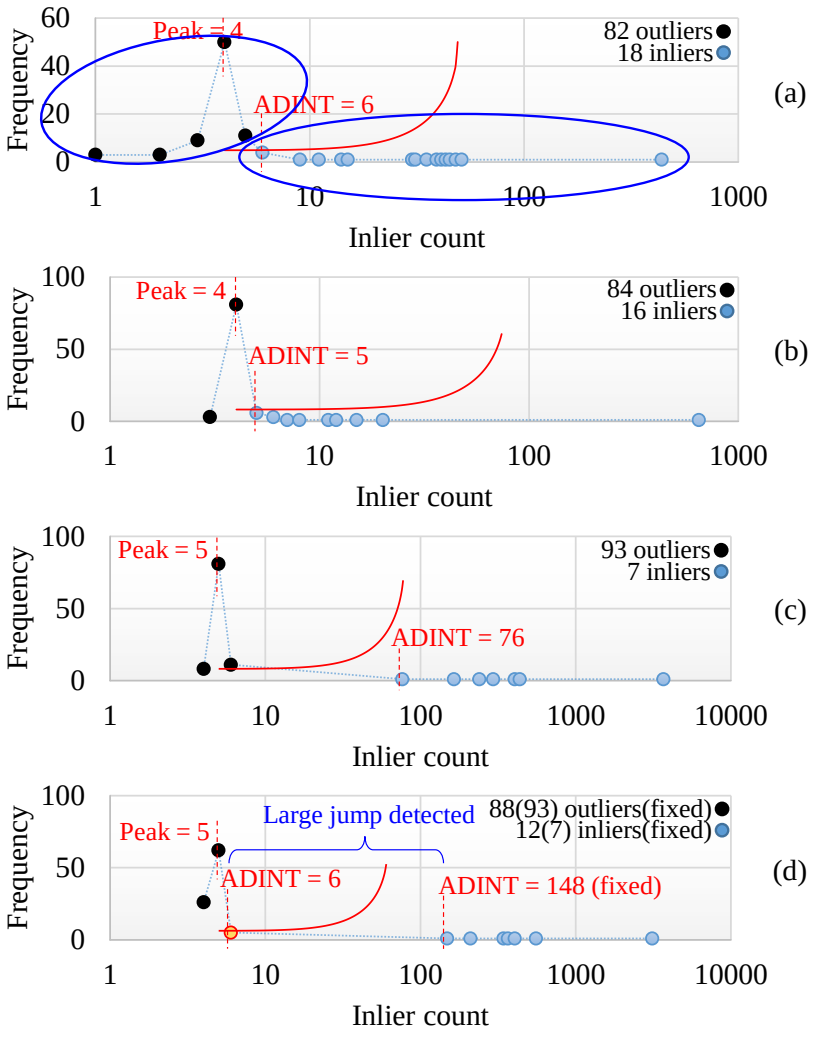


FIGURE 4.8: The actual cases of how ADINT finds an inlier threshold. The dots represents the frequency of each inlier count (shown in log scale). The ADINT ratio was set to 0.9 (red arc). (a) Blue ellipses show the group of outlier images (left) and inlier images (right). (b,c) General cases. (d) Special case when ADINT finds a threshold at the boundary of the outlier group (red-circle dot), which is then fixed to the inlier group.

4.3.2 Method

Here, we propose an adaptive *inlier* threshold algorithm, namely, ADINT that adaptively determines *inlier* threshold to filter relevant images. First, we use LO-RANSAC to determine which images in the top-k retrieved images are under a certain inlier count threshold. Then we construct a frequency histogram (the number of images) of the inlier count to be processed with the rest of the ADINT algorithm. The example of histograms are shown in figure 4.8(a)-(b). Generally, images with high inlier counts tend to be the correct matches (see the right ellipse in figure 4.8(a)), while images with low inlier counts tend to be outliers (see the left ellipse in figure 4.8(a)). More specifically, outlier images are usually at a peak on the left side of the distribution, including its neighborhoods and

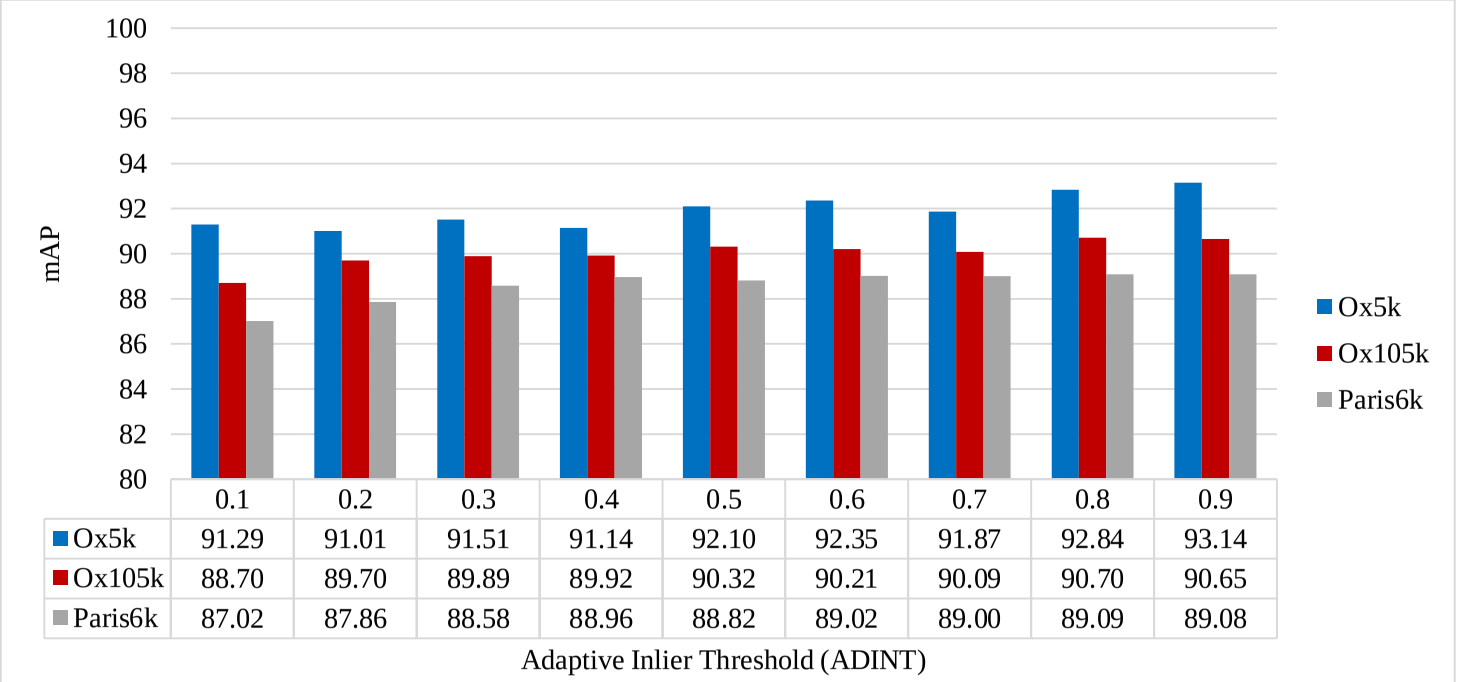
the images that have lower inlier counts than the peak point. Based on this observation, our ADINT finds a splitting point for inlier counts as follow:

Algorithm 2 Adaptive Inlier Threshold (ADINT)

Require: $inls$ (as inlier count distribution)
1: **procedure** ADINT
2: $center \leftarrow find\_max(inls)$
3: $adint_{ratio} \leftarrow predefined$
4: $inl \leftarrow inls[center * adint_{ratio}]$
5: *loop*: $center.idx < inl.idx$
6: $a \leftarrow center - inl$
7: $b \leftarrow |center.idx - inl.idx|$
8: $c \leftarrow \sqrt{a^2 + b^2}$
9: **if** $center * adint_{ratio} \leq c$ **then**
10: $adint = inl$
11: **if** $find\_large\_jump\_ahead(inls, adint) \neq null$ **then**
12: $adint = find\_large\_jump\_ahead(inls, adint)$
13: **stop**;
14: $inl \leftarrow inls[inl.idx - 1]$
15: **return** $adint$

In order to determine the inlier threshold between inlier and outlier images, we need to feed the top-k images to LO-RANSAC to obtain the raw inlier count as shown in figure 4.7, then we build an inlier histogram as shown in figure 4.8. The algorithm 2 is then aim to find the splitting point bewteen into two inlier groups, which are a group of high inlier (as blue dots) that tend to belonging to inlier, and a group of low inlier (as black dots) that is definitely belonging to outlier. To split it, the algorithm firstly finds a point in a histogram where the most images are belonging to as a pivot. This pivot point will be a center point for the following sweeping mechanism. The algorithm determines the inlier threshold by sweeping clockwise from the right-most inlier count that can be reached by the radius of 0.9 times the frequency of the center point. The sweeping process will be done at the point where the radius cuts the histogram, or until it reaches the inlier count of the center point. Hence, the first point on an inlier histogram that far beyond this sweeping will be an Adaptive Inlier Threshold. The ratio is introduced, namely, ADINT ratio, with a typical value of 0.9 for controlling the radius. On the reason why ADINT ratio for sweeping the inlier count has to be 0.9 in our experiments is that, We did several trials with this ratio from 0.1 to 0.9 as reported in figure 4.9, And we found that 0.9 is the most appropriate value that agreed by all datasets to achieve the best performance.

ADINT works well in most cases. However, when there is only few correct matches in the retrieved images, we might easily identify inlier counts into two distinct group (see black and blue points in figure 4.8(c) and (d)). On the other hands, setting an ADINT ratio



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Ox5k	91.29	91.01	91.51	91.14	92.10	92.35	91.87	92.84	93.14
Ox105k	88.70	89.70	89.89	89.92	90.32	90.21	90.09	90.70	90.65
Paris6k	87.02	87.86	88.58	88.96	88.82	89.02	89.00	89.09	89.08

FIGURE 4.9: The figure shows several experiments are done on different ADINT ratio on different dataset. The plot shows that ADINT ratio of 0.9 is the most appropriate value that agreed by all datasets to achieve the best performance.

to 0.9 might not be enough, and may fail to filter out some irrelevant images as seen in figure 4.8(d). Therefore, we check whether a threshold was found near a boundary of an outlier groups, and if so, the threshold will be fixed to a boundary of an inlier group (see algorithm 2 at line 10 and 11). Figure 4.10 compares the auto-selected inlier images with the ground truth images found through the retrieved images, by selecting only one query topic as a sample to show our ADINT result of relevant images selection. The figure 4.10 represents the highly ranked images obtained from the first round retrieval. A blue line shows an inlier count that obtained from a LO-RANSAC, while a red line shows our ADINT inlier threshold. For any image that contains inlier count (blue line) higher than ADINT threshold (red line) will be marked as selected (orange). As result, our selected images are somehow mostly correct comparing to the ground truth as highlighted with gray bars.

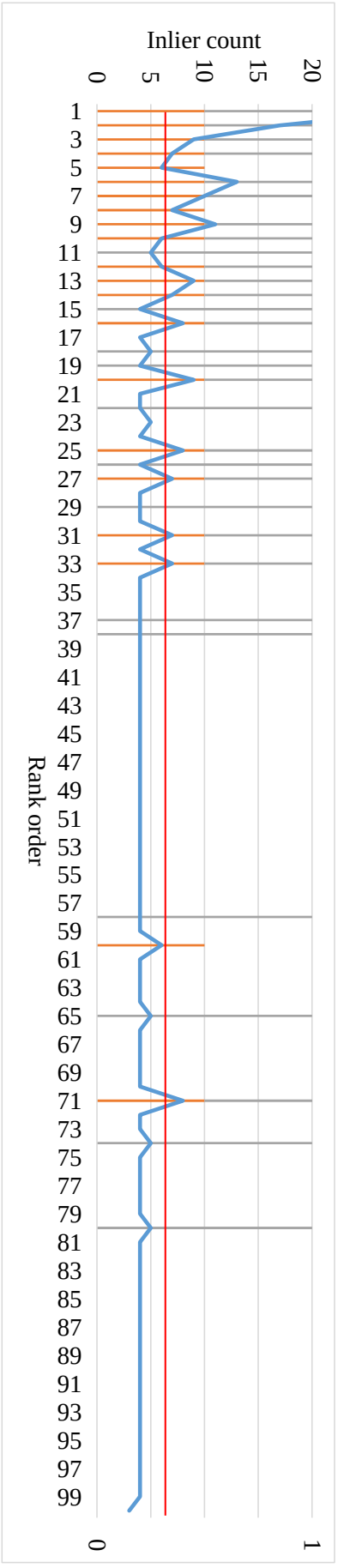


FIGURE 4.10: The blue line shows the total number of inliers found in the top 100 images of the initial rank on the sample query with a baseline AP of 30.14%. The gray bar shows the true positive images appearing in the ranked list, whereas the orange bar shows images selected by our adaptive inlier threshold (ADINT) seen in a red line that yielded a final AP of 82.07%.

Inlier Threshold	AQE (mAP %)			QB + SP (mAP %)		
	Ox5k	Ox105k	Paris6k	Ox5k	Ox105k	Paris6k
3	88.11	79.69	80.44	74.39	50.95	89.66
5	88.60	80.72	80.13	85.47	68.44	89.32
7	87.87	81.86	79.19	92.48	89.31	87.76
9	87.32	81.15	78.87	91.64	88.28	86.62
11	87.13	80.85	78.70	90.77	87.56	85.88
A	**87.88**	**81.85**	**78.70**	**93.49**	**90.36**	**88.96**
Δ(min, **A**)	0.75	2.16	0.00	19.10	39.41	3.08
Δ(max, **A**)	-0.72	-0.01	-1.74	1.01	1.05	-0.70

TABLE 4.1: Comparison of fixed *inlier threshold* and adaptive inlier threshold (ADINT, **A**). QB + SP performed the best with ADINT, and ADINT provided a fair performance for AQE.

4.3.3 Evaluation

We compared the effect of the adaptive inlier threshold algorithm (ADINT) with that of a fixed inliner threshold (FINT). Here we are testing on an Adaptive Inlier Threshold algorithm, which is applicable to only SP based method like AQE, and QBSP. So, for fairness, we compared only QBSP and AQE. To be fair evaluation, we also let AQE use our ADINT as well as a traditionally fixed inlier threshold (FINT). The results are shown in table 4.1. AQE had less impact on the various inlier thresholds, while QB + SP had a strong impact. As table, the different between *min* and *A* is how much ADINT better than minimum of FINT, and different between *max* and *A* is how much ADINT better than maximum of FINT. The summary of the result is that, ADINT is better than FINT in most cases of QBSP, so ADINT significantly improved the performance of QBSP, And ADINT is on par to AQE, so its contribution to AQE was minor. At least, our parameter tuning is an automated threshold, that is what we really proud of our newly contributed algorithm.

4.4 Results

Bringing back the spatial verification method like LO-RANSAC on top of our proposed QB framework, which named as QBSP, shows the prominence result of final retrieval performance Additionally. as QB fails on some query that does not contain sufficient true positives (see figure 4.11), which in total top-k images were dominated by irrelevant images as described in section 3.5, while QB + SP does much better result on this problem. The overall mAP of QB + SP are reported in the evaluation chapter 6.

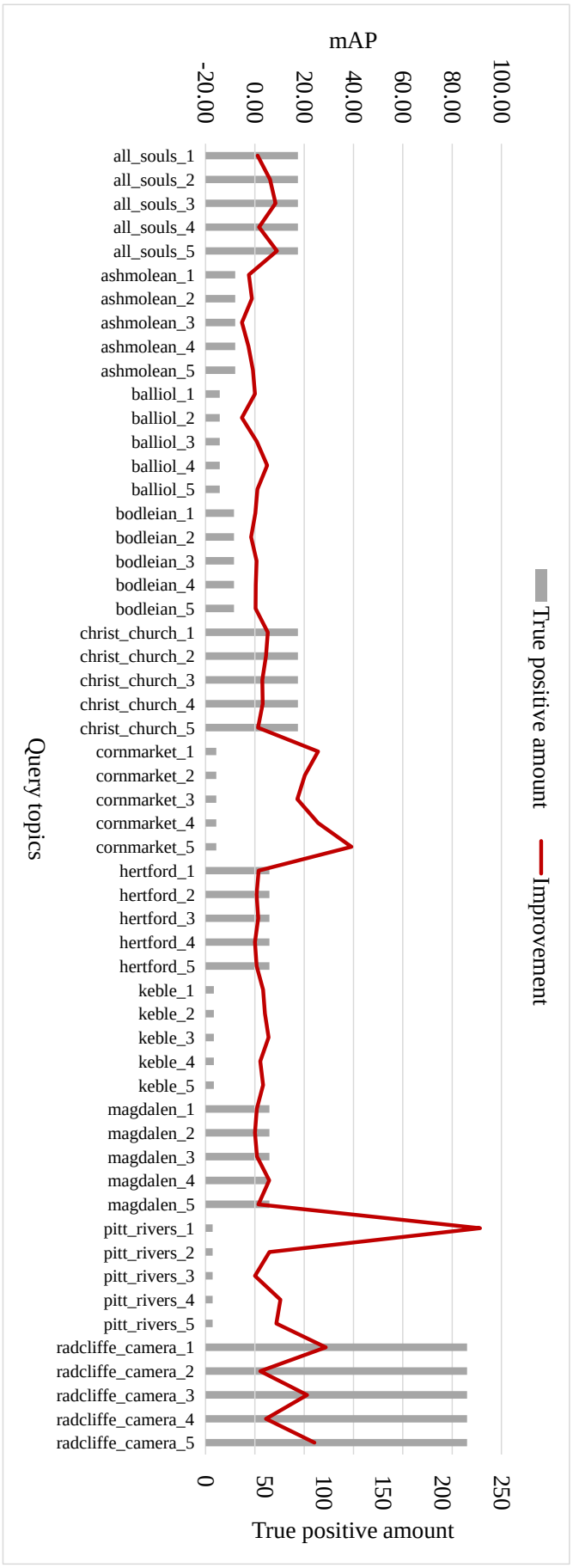


FIGURE 4.11: The relative improvement for each query topic of QB + SP over QB on the Oxford 5k dataset show QB + SP improve the retrieval performance much better than QB on the low number of true positive list.

Chapter 5

Speed-up mining process

"It does not matter how slowly you go as long as you do not stop."

— Confucius

Once we do mining on a top-k relevant images with a very large vocabulary size, we found the patterns may response to the visually consistent objects on many images. This creates too large permutation of the possible patterns into the mining space, which is a direct caused of a time consuming problem for the mining component of our QB.

Thereby, we aim to reduce the time consumption for the pattern mining, since the original FIM tool was designed to handle a small number of item while having large number of transactions is not a problem. In our task, we employed FIM for using with a visual words, which we originally used up to 1 million words as our vocabulary size. The total number of item in this case is the total non-zero words from the BoVW histogram, and the number of transaction is the number of top-k images. Hence, the number of item can be too large, or even reaching to 1 million is still harder but possible. In this chapter, we present the way to speed-up a mining process by using a technique from a data mining field. Also, we evaluate the time consumption used by this technique, which shows significantly improved of speed comparing to the traditional mining approach.

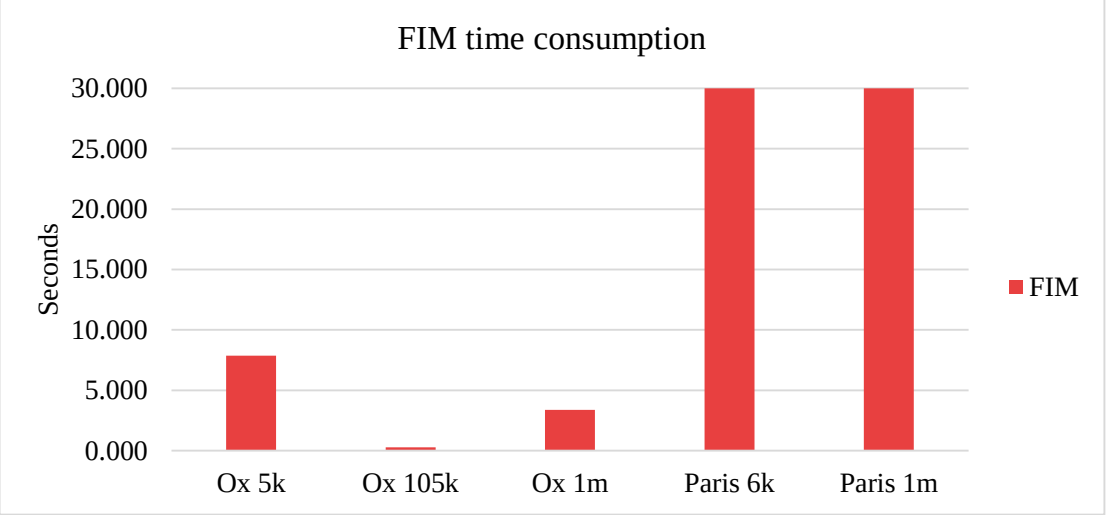


FIGURE 5.1: The evaluated time consumption taken by all dataset includes Oxford 5k, 105k, 1m, Paris 6k and Paris 1m.

5.1 Motivation

We presented QB and further boosted the performance by using a spatial verification as QBSP, with the integration of automatic parameter tuning algorithms. However, we found that our proposed QB and QBSP seemed to be slower than the other approaches. So, we evaluate the time consumption of FIM taken by several approaches and also report the result in the following figure 5.1.

We ran these experiments using the parameters finely tunned for achieving the best performance for each of the methods. (the best parameters configuration are described in the session 6.7). In a real-time object-based image retrieval task, waiting for result within a few second is generally fine. However, as the result reported in the figure 5.1, show we may need to wait for searching result from 5 to 30 second on average caused by the employed FIM process within QB and QBSP. So, this amount of time is rather large, and not rated as a real-time practical retrieval system. By this reason, we additionally focus on the way to improve the speed of FIM by using several trick on the practical implementations, the way to feed image transactions to FIM, and the way to import such large number of the patterns from the FIM output. Although, there are several implementation tricks were actually implemented in this work, however, in this chapter, we will discuss more in detail of a technique that lead to a very big jump which reduce a lot of time usage by FIM. The technique is called as a transaction transposition.

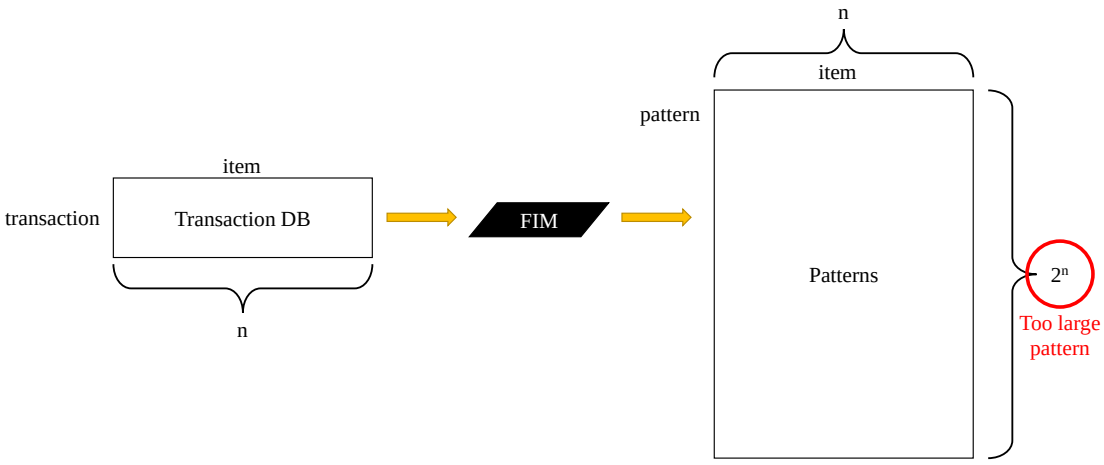


FIGURE 5.2: The figure of pattern discovery using a normal transactions database with very large number items as possible to 1 million item (visual words). By using a standard way of pattern mining, the final pattern is usually large as 2^n when n is a number of items. So, it can be too large as an intangible problem.

5.2 Transaction transposition

Generally, FIM tools are designed for a very large number of transaction and not much number of different item, according to the market basket analysis purpose. FIM usually find the number of pattern up to 2^n as n is the total number of identical item from several baskets. Contrastingly, in our case, we use a very large visual words as 1 million words to build the vocabulary for images. So, FIM task is to discover on a very high number of pattern up to 2^n, where n is the non-zero word identically collected from many images, which can be rather huge as illustrated in figure 5.2.

The theoretically method called transaction transposition helps reducing the total mining space, which then map the mining result from the transposed transaction back to the original aspect using a Galois connection [146] through an inverse relationship on a complete lattice. By doing a this, we can save a lot of mining time using on any FIM algorithms. there is a theoretically method called transaction transposition This technique can also help speed-up the mining process, as employed in several work e.g., Voravuthikunchai et al. [144], Jeudy and Rioult [147], Joshi and Jain [148]. The useful of this technique is to "transposed" the transaction database from a very large column to be significantly less number of columns, and the row of transaction also be switched from the original one. The illustration of this technique can be seen in figure 5.3.

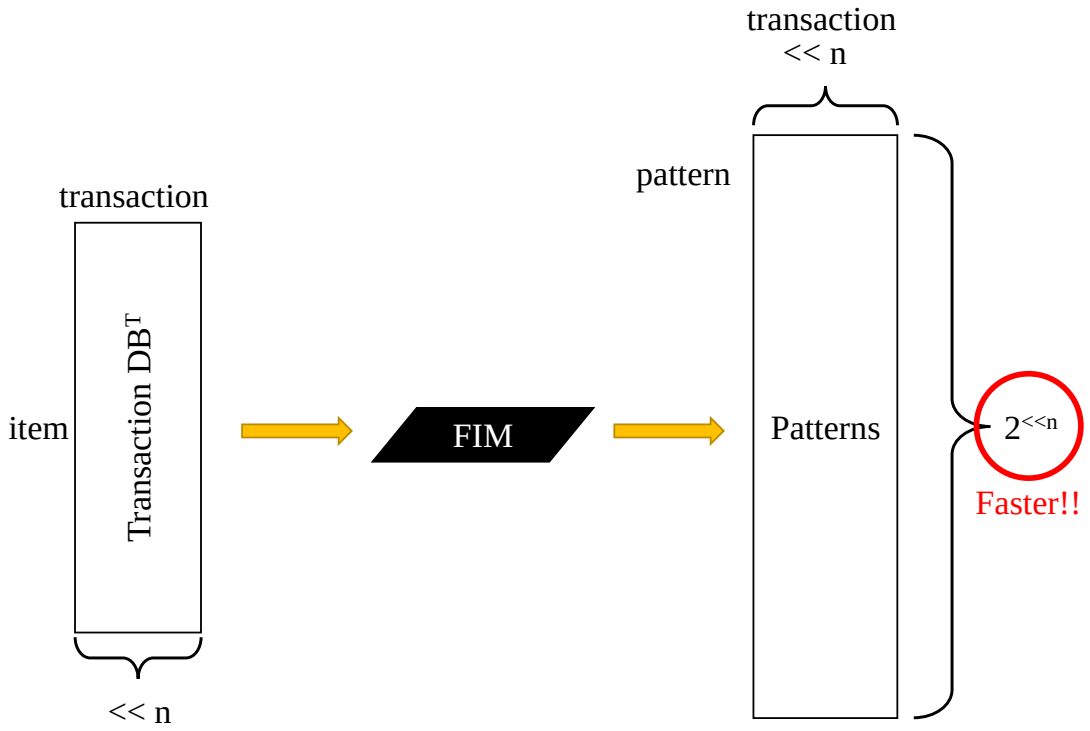


FIGURE 5.3: By transposing a transaction database of the figure 5.2, this figure illustrates a pattern discovery using transaction transposition technique, which will produce the pattern from much smaller item amount, as from n item to only the number of transaction or top-k images, or the total number of patterns to be discovered in the mining space will be 2^k that is much less than 2^n where $k \ll n$.

There are several interesting questions which is raised when doing a transaction transposition:

1. What kind of information can be gathered in the "transposed" transaction database on the patterns of the original transaction database?
2. How can we regenerate the patterns in the original transaction database from the patterns extracted in the "transposed" transaction database?
3. How can we generate all the itemsets satisfying a constraint using the extracted of such patterns.

Img. I_k	Trans. t_k
I_1	$t_1 = \{i_1, i_2, i_4, i_6\}$
I_2	$t_2 = \{i_2, i_5, i_8\}$
I_3	$t_3 = \{i_2, i_3, i_9\}$
I_4	$t_4 = \{i_1, i_2, i_4, i_7\}$
I_5	$t_5 = \{i_2, i_3, i_8\}$

TABLE 5.1: A simple transaction database of top-five images.

5.3 Usage

Given transactions with m items, FIM generally discovers through shared patterns from 2^m possible subsets of the permutation pool generated by items of a transaction database. For a toy example, suppose we have three items or let's say code words of a, b, and c; the total number of possible patterns in this pool will be $2^3 = 8$, that actually consisted of the complete set of patterns as $P \in \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$.

According to our problem, since we use 1 million words as visual vocabulary for building BoVW, these BoVW vectors are rather high dimension, and the sparsity is also quite high. For instance, let us assume that an image has been encoded into a BoVW histogram with 1000 non-zero words. Suppose further that we have ten images in a ranked list that have 200 words in common while each of the other 800 words appear only once in them. In total, there would be $200 + (800 \times 10)$, or 8200 distinct words to be handled by FIM. Therefore, $m = 8200$ yields $2^{8200} \cong 2.79 \times 10^{2468}$ possible sets, which is really large size makes the search space intractably large for FIM.

Although the number of items can be large (in our case several thousands), the number of transactions is relatively small (a few tens to hundreds). A number of pattern discovery studies [149–151] have tried to reduce a dimension of n, by using a Galois connection to do mapping on a completed lattice from a transposed matrix side back to the original untransposed matrix side. Before doing a transaction transposition, we need to check whether our case satisfying the condition said in Galois mapping as follow:

> "The mapping f is antitone and there exists an antitone mapping g from P to P' such that the composition mapping are extensive.– (GM)[146]"

Let P and P' are two complete lattices generated from a transaction database T and a transposed of a transaction database T is T^T. Using our toy example on a table 5.1, we then found total patterns of P and P' are isomorphic to each other on a complete lattice as shown in a figure 5.4.

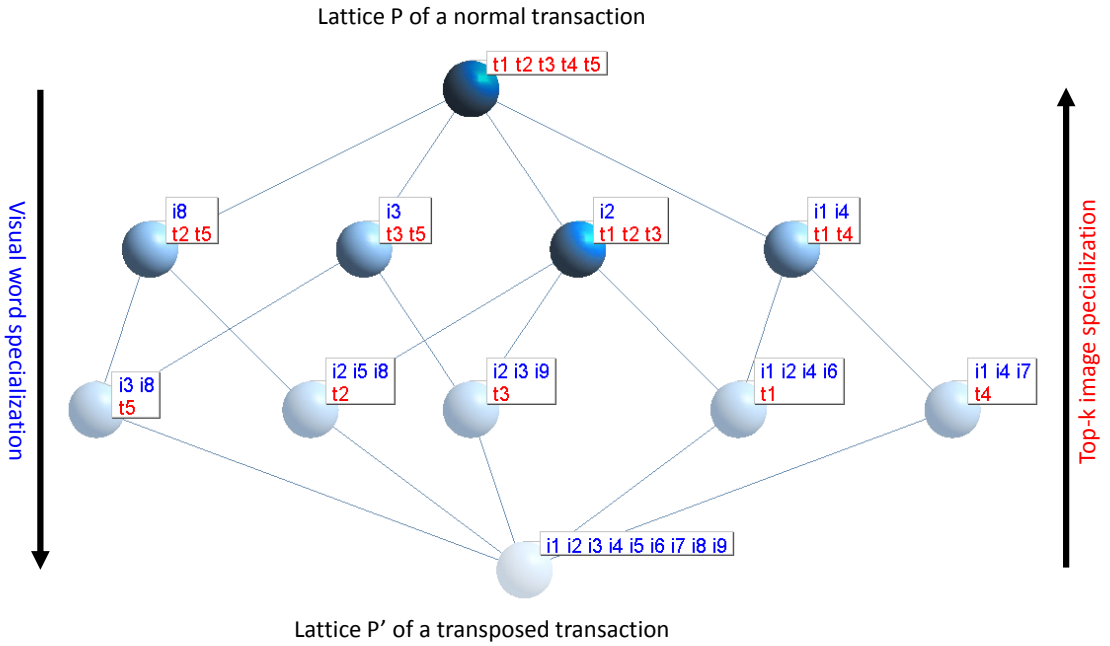


FIGURE 5.4: Two complete lattices[a] of (top-down) a toy example transaction database and (bottom-up) a transposed transaction show the isomorphic property which satisfy the Galois mapping condition.

[a]The lattices of this toy example is visualized by Lattice Miner. [152]

Briefly speaking, transposing a transaction database, which contains m items with n transactions where $m \gg n$, will let FIM discovers on much smaller amount of the possible patterns pool as the total possible patterns are only 2^n, which is $\ll 2^m$. Therefore, FIM will operate much faster than using an original transactions. The patterns discovered from the original transaction database and the transposed transaction database are visualized in a figure 5.5. Also, a timing report of both version $FIM(s)$ and $FIMT(s)$ are shown in a figure 5.6).

As we found out our target patterns can be mined from T and will be faster with T^T, however, the meaning of both pattern results are different. To be able to map from the transposed transactions space to the original transactions space, we will discuss about the meaning of each side for more understanding on how to map these two space back and forth.

Original space Mining patterns from original transaction database means, *we are finding which visual word sets shared among images*, where $p \in P : p = \{i_1, i_2, i_3 ... i_m\}$.

Transposed space Mining patterns from a transposed transaction means, *we are finding which images contain similar visual words.*, where $p' \in P' : p' = \{t_1, t_2, t_3 ... t_{k'}\}$.

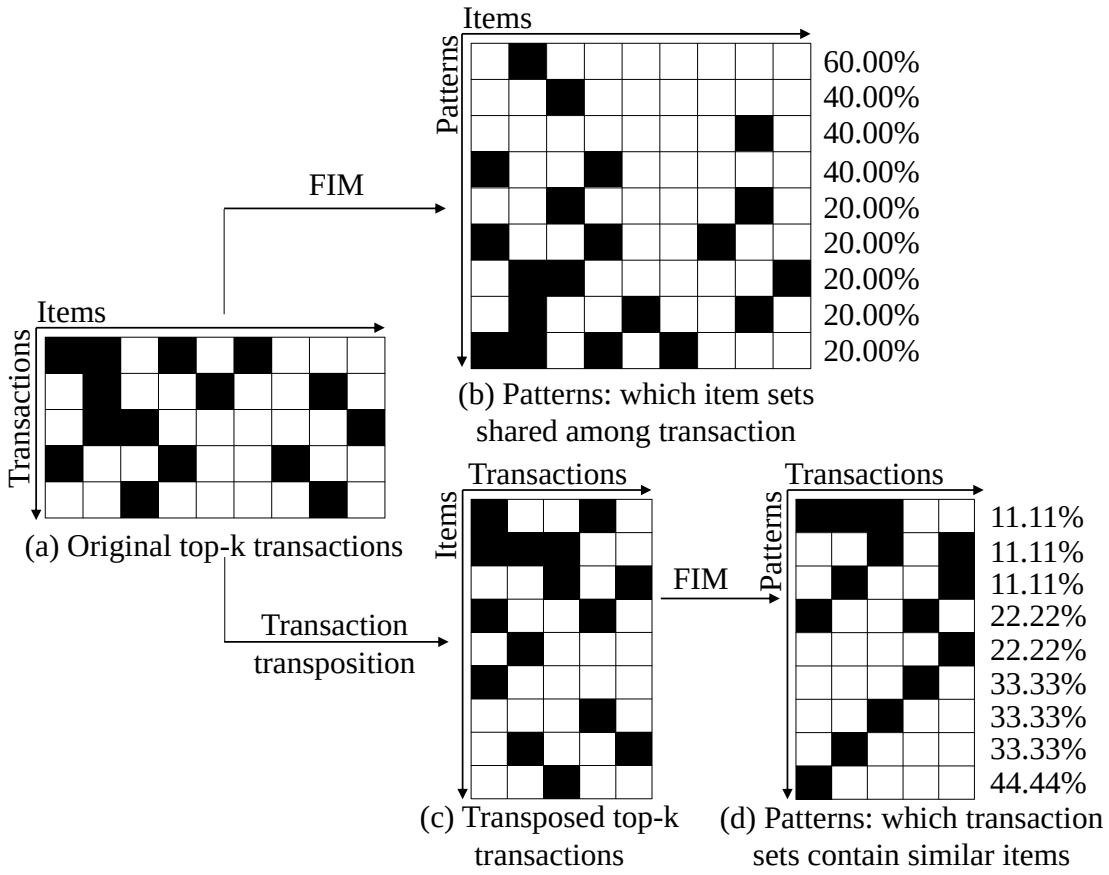


FIGURE 5.5: (a) Original transaction database (T). (c) Transposed transaction database (T^T) for speeding up FIM (b), (d) Patterns (P and P') discovered by using FIM on a normal database and a transposed database.

In order to utilize a patterns P', we need a mapping function $f : P' \rightarrow P$ as follow:

$$f(P') = \forall p'_{j',t'} \in P' : \mathbf{A} \tag{5.1}$$

where

$$\mathbf{A} = \bigwedge \begin{bmatrix} [T(p'_{1,1}) \wedge T(p'_{1,2}) \wedge \ldots \wedge T(p'_{1,k'})]_1 \\ [T(p'_{2,1}) \wedge T(p'_{2,2}) \wedge \ldots \wedge T(p'_{2,k'})]_2 \\ \vdots \\ [T(p'_{j',1}) \wedge T(p'_{j',2}) \wedge \ldots \wedge T(p'_{j',k'})]_j \end{bmatrix} \tag{5.2}$$

and

$$\mathbf{A} = P \tag{5.3}$$

where P' will be map to P through a transaction T, $T(x)$ will return a set of items on original T, and the total number of patterns on both space will be the same, as $\|P'\| = \|P\|$ or $j' = j$.

To be more clear on how this function can map two space together is that, from the patterns P', we map back each item founded in p', which corresponds to a *transaction id* t', to an original transaction database T. The actual set of items i_m on each mapped transaction will be checked to find which item appear on all transactions t'. And such item i_m will be collected to build p as a mapped pattern from $p' \rightarrow p$. In the final sense, we will discover patterns several order of magnitudes faster than a traditional way.

5.4 Evaluation

We then evaluate the time usage for both FIM and $FIMT$ on the standard evaluation datasets including our extended distractor datasets. The timing report on both FIM and $FIMT$ are shown in the following figure 5.6. As result, $FIMT$ can significantly reduced the time usage of a mining process on all datasets. And also, we can easily regenerate the result in the original aspect by using the result of the transposed version within less amount of time.

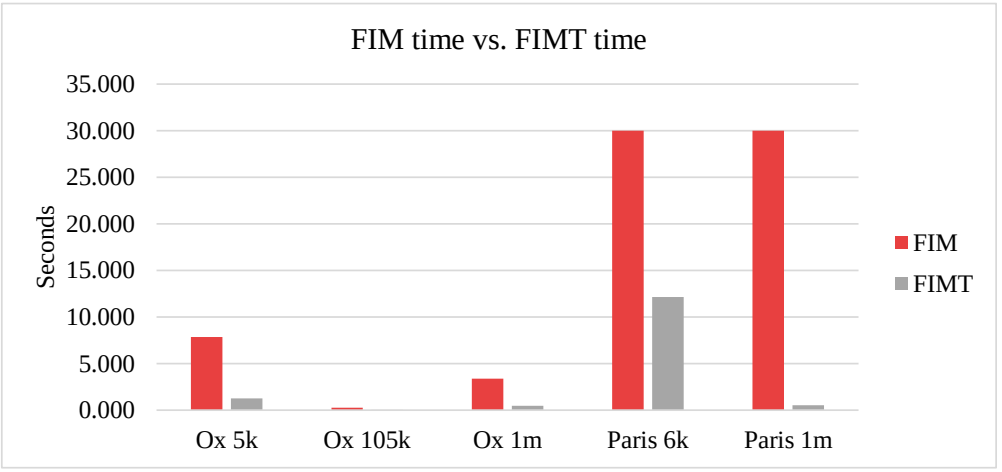


FIGURE 5.6: The figure show an average time usage when using FIM on a standard transaction database comparing to the transposed version as FIMT on a transposed of transaction database.

Chapter 6

Experimental setup, evaluations, and discussion

"Success is not final, failure is not fatal: it is the courage to continue that counts."
— Winston S. Churchill

We proposed the works for help improving the retrieval performance by derive the main benefit of frequent itemsets mining technique for finding co-occurrences of object pattern through the first round relevant images, which results to relax the constraints from using a query image. In this chapter, we evaluate the retrieval performance on both standard query and the degraded query. We compare our works QB and QBSP with our re-implemented of a standard BoVW follow the framework of Zhu and Satoh [73], Zhu et al. [78, 122], the standard average query expansion (AQE[32]) by Chum et al. [32], our implemented of average query expansion (AQE), our bare mining work as a query bootstrapping (QB) with the automatic support tuning algorithm (ASUP), and our QB work with spatial verification (QB + SP) and together with the adaptive inlier threshold tuning algorithm (ADINT). As result our methods show very impressive result that has significant improvement over the selected baselines. Also, we also discuss about pros and cons with the limitation, the target datasets, and the requirements of using our works for better understanding and for integrating our work in a proper way.

6.1 Datasets and evaluation protocol

The dataset used in this research are well known which firstly introduced in the work of Philbin et al. [22, 37] to be the standard datasets of object based image retrieval, namely, Oxford 5k, Oxford 105k, and Paris 6k as targeted for a landmark retrieval. Together that, they provide the ground truth sets representing the relevant images corresponding to each of specific query topic. Moreover, we evaluate the retrieval performance by using mean average precision (mAP) as an evaluation metric for all of the experiment running on this work. For example, one run means to retrieve the result for the provided 55 queries, all these results will be calculated the average precision (AP) comparing to the ground truth list. Therefore, mean average precision is calculated by averaging all the APs together as an mAP for one run. The datasets description is describe as follow:

The Oxford Buildings 5k The dataset consists of 5,062 high-resolution images that were crawled from Flickr [2] by using queries of the famous Oxford landmarks, such as "Oxford Christ Church" and "Oxford Radcliffe Camera".

The Oxford Buildings 105k The dataset is extended from an Oxford 5k dataset by including 100k of distractor images from the Flickr 100k dataset, as it originally consists of 100,071 images, which collected by searching for popular Flickr tags.

The Paris 6k The dataset consists of 6,412 high-resolution images collected from Flickr by searching for particular Paris landmarks, such as "Paris Moulin Rouge" and "Paris Arc de Triomphe".

The Ground truth labels The ground truth is provided for an Oxford dataset and a Paris dataset independently. Each of dataset is provided with 11 different landmarks, that consisting of 5 variances view angle per a landmark. Combining of these set, the dataset provide the set of 55 query topics in total, with four possible labels as follows:

1. *Good* – a clear picture of the object/building.
2. *OK* – more than 25% of the object is clearly visible.
3. *Bad* – the object is not present.
4. *Junk* – less than 25% of the object is visible, or there is a very high level of occlusion or distortion.

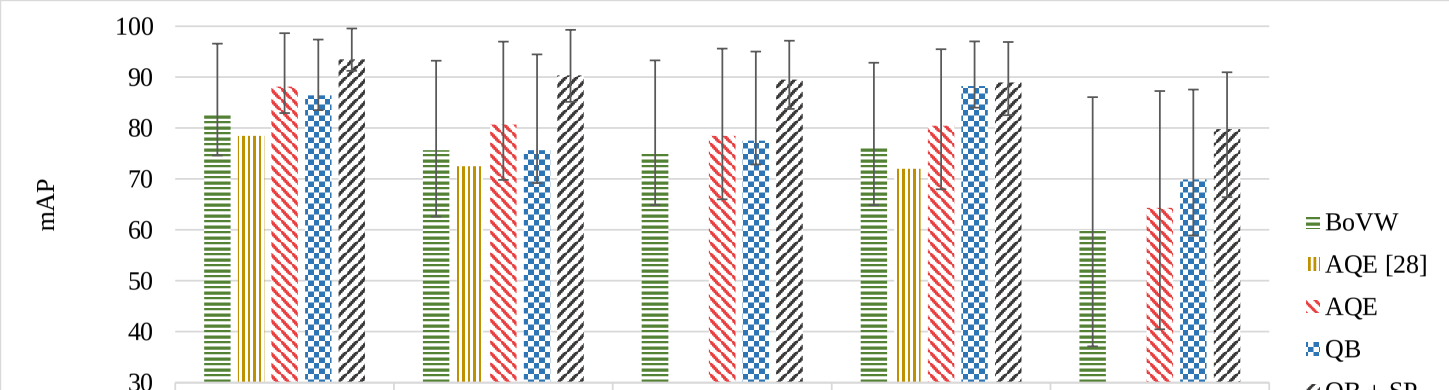
Name	Total images	Total features	Sampling	Cluster
Stanford-MVS	1,193	1,458,667	All	1M
Oxford 5k	5,063	17,390,270	10M	1M
Oxford 105k	105,134	329,147,561	10M	1M
Paris 6k	6,392	20,244,882	10M	1M
Oxford 1M	1,005,063	798,393,536	100M	1M
Paris 1M	1,006,392	801,248,148	100M	1M
INS 2011	1,650,827	917,656,466	200M	1M
INS 2012	2,256,930	1,908,832,917	200M	1M
INS 2013-2014	7,837,877	9,814,391,541	200M	1M

TABLE 6.1: The statistic shows sampling rate for building a codebook.

6.2 System and parameters configurations

In this section we describe on the configurations and the parameters that we applied through all this research. Beginning from the feature extraction, we use 128-dimension of SIFT feature detector with Hessian Affine descriptor from the implementation of Michal Perďoch [15, 16]. Also, we modified the code to produce the descriptor with suggested namely as a RootedSIFT by Arandjelovic [30]. For the encoding step, we used randomly sampling from the local features on each of dataset independently, as reported in the table 6.1, for training the a codebook with the vocabulary size of 1 million words. In particular, we used an approximate k-mean algorithm (AKM) to build up the clusters by using a FastCluster by Philbin et al. [22, 23], with 50 iterations for terminating the clustering process. The visual word assignment was done with an approximate nearest neighbor search algorithm (ANN) by using a fast library for approximate nearest neighbor (FLANN) by Muja and Lowe [24], Muja [25]. We use a hard assignment for nearest neighbor, 512 checks on maximum leafs on the randomized KDTree index. For the spatial verification, we used a LO-RANSAC library by Lebeda et al. [33] with the default parameters. However, we increased the random samples from 1000 to 3000 with the expect for obtaining a small accuracy improvement. Moreover, addition to the parameters indicated above, we enabled the local optimization option (enable lo=true) for boosting up the performance as claimed in his paper.

We conducted all experiments on a computer running Linux RHEL 6.3 with an Intel Xeon E7-4870 @2.4 GHz having 40 virtual CPUs. For time usage evaluations, we evaluate all the algorithm based on pre-cached of an inverted index database with 100% cache hit on our machine with the maximum of 512 GBs of memory. The core retrieval module was written with the compliance of the C++11 standard, and all the codes were compiled with the optimization -O3 flag using cross-compiled GCC 4.8.1. We evaluated the time usage of FIM tool and the similarity scoring process based on a single CPU. Although, since we parallelized RANSAC process, the time usage reported in this paper are conducted with of the available CPUs.



	Ox 5k	Ox 105k	Ox 1m	Paris 6k	Paris 1m
BoVW	82.84	75.66	75.28	76.33	59.95
AQE [28]	78.50	72.50		72.00	
AQE	88.12	80.71	78.48	80.44	64.32
QB	86.41	75.67	77.56	88.28	69.94
QB + SP	**93.49**	**90.36**	**89.52**	**88.96**	**79.81**

FIGURE 6.1: Comparison of (*QB* and *QB + SP*) with BoVW, AQE, and AQE [32] on Oxford 5k, Oxford 105k, and Paris 6k. QB and QB + SP significantly outperformed the other methods.

6.3 The overall comparison

The overall retrieval performance for all methods were done on its best parameters configuration as we finely tune in details. However, for the QE based approaches (AQE, AQE[32], QB, and QB + SP), setting various top-k values will directly effect to the final performance. We ran the top-k experiments using the optimal parameter configurations as finely tuned in details for the best performance on each method independently Shortly, we used the top-100 images for AQE, QB + SP on all datasets, the top-100 for QB on the Paris dataset and the top-25 for QB on the Oxford datasets (The details of top-k tuning will be discussed in section 6.4). However, for the AQE[32], due to a speed issue, top-10 images is only the maximum for them.

Figure 6.1 shows the performance of BoVW, AQE[32], AQE, QB, and QB + SP respectively. The result is clearly explained itself from this figure that QB + SP significantly outperformed the other methods and also show the improvement better than AQE. In general, QB + SP shows the highest AP in all queries, which is implied that the overall recall also got improved. Moreover, our QB + SP also shows the significant improvement from the standard BoVW baseline on the very high distractor images as our newly extended datasets e.g., Oxford 1M and Paris 1M.

The retrieved result sample are shown in the selected result in figure 6.2. Also, the figure 6.2 shows the top retrieved images have better visual relevancy by AQE and much better with our QB + SP comparing to the standard BoVW baseline. Additionally, AQE improve the result (figure 6.2d) to be closer to the provided query region (figure 6.2a), and QB + SP improve the result (figure 6.2f) to be closer among the relevance images found in the retrieved list. So, this is the main evidence that our methods already relax the spatial constraint from the query image. However, our method will improve the retrieval performance if the standard BoVW provides some good enough result. Hence, it has less problem to the different query images.

(A) Query

(B) Baseline results (top-20).

(C) True positives of a baseline (top-100).

(D) AQE results (top-20).

(E) True positives of AQE (top-100).

(F) QB + SP results (top-20).

(G) True positives of QB with LO-RANSAC spatial verification (top-100).

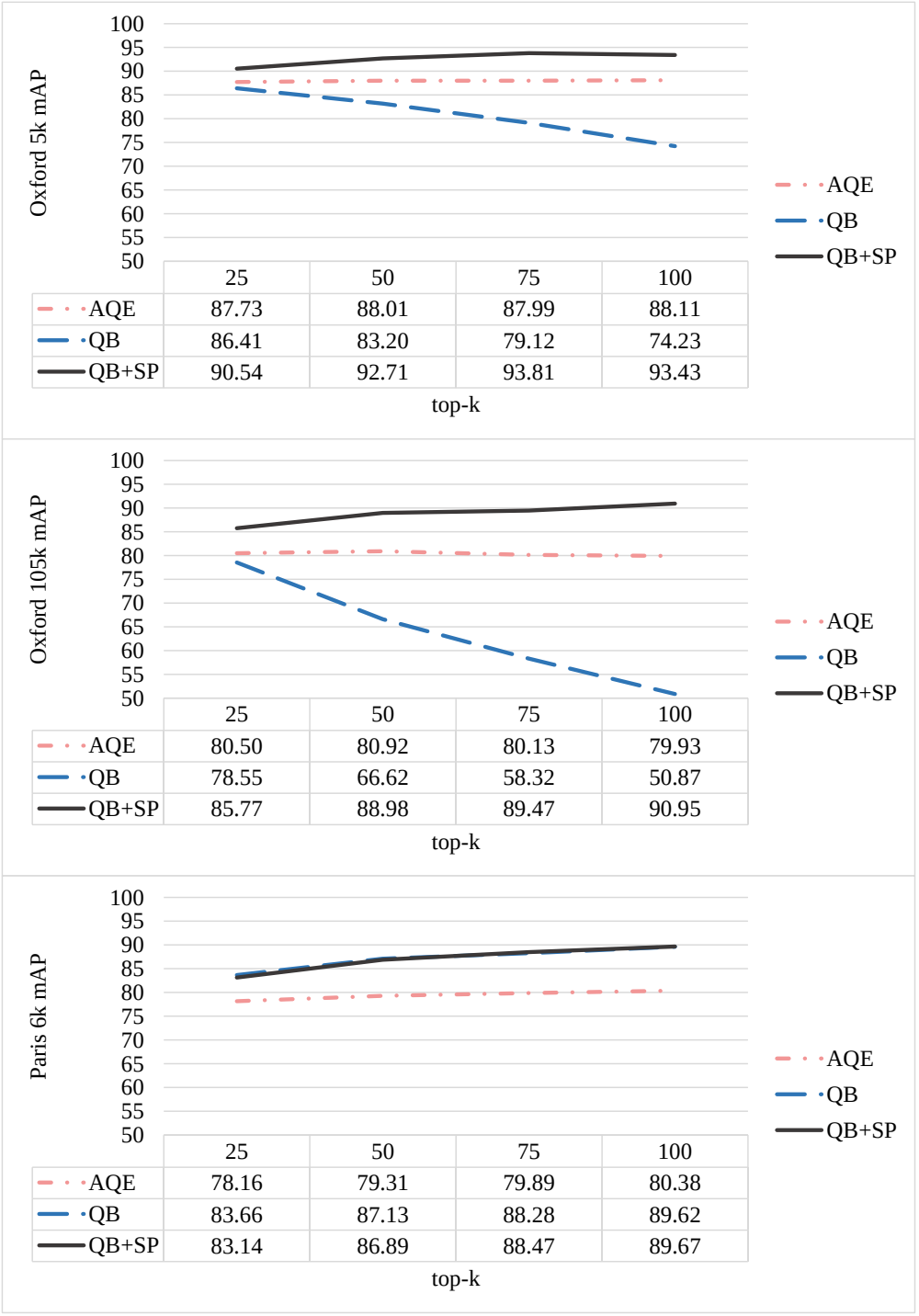
FIGURE 6.2: Top 20 relevant images showing how the three main approaches differ in effect and the corresponding true positive lists. (A) Query image with ROI (B-C) BoVW result with an AP of 78.82% (D-E) AQE got better matches to ROI than BoVW with an AP of 94.44% (F-G) QB + SP reflects the frequent objects from the initial rank with an AP of 99.76%.

6.4 Impact of the number of relevant images to retrieval performance

One of the parameters of query expansion based methods (e.g., AQE, QB, and QB + SP) is the number of relevant images k from the first-round retrieval. We varied k (e.g., , 25, 50, 75, and 100) to see how it affected the retrieval performance.

As figure 6.3 shows, the spatial verification based methods (AQE and QB + SP) received a positive impact for a larger number of images, while those without spatial verification (QB) suffered when too many images were added. A possible reason for the non-spatial verification methods doing less well is that a larger number of images may have more irrelevant images; spatial verification eliminates such irrelevant images. In addition, QB + SP takes full advantage of the more numerous relevant images because it can utilize the consistency among them. In contrast, AQE did not enjoy this benefit because it imposes pairwise consistency between the query and each of the images in the ranked list.

Figure 6.3 also shows that increasing the number of images had a negative impact on the recall of QB with Oxford 5k and 105k, but a positive impact with Paris 6k. This difference is due to the different numbers of relevant images per query. The number of true positive images per query of the Paris 6k dataset is on average 163 (minimum 51 and maximum 289). On the other hand, Oxford 5k and 105k datasets contains on average only 51 true positive images per query (minimum 6 and maximum 221). This is why QB did not degrade the recall on Paris 6k even with up to 100 images.



	25	50	75	100
AQE	87.73	88.01	87.99	88.11
QB	86.41	83.20	79.12	74.23
QB+SP	90.54	92.71	93.81	93.43

	25	50	75	100
AQE	80.50	80.92	80.13	79.93
QB	78.55	66.62	58.32	50.87
QB+SP	85.77	88.98	89.47	90.95

	25	50	75	100
AQE	78.16	79.31	79.89	80.38
QB	83.66	87.13	88.28	89.62
QB+SP	83.14	86.89	88.47	89.67

FIGURE 6.3: Impact of different top-k values on the retrieval performance of the query expansion based method.

6.5 Automatic parameters and relative improvement

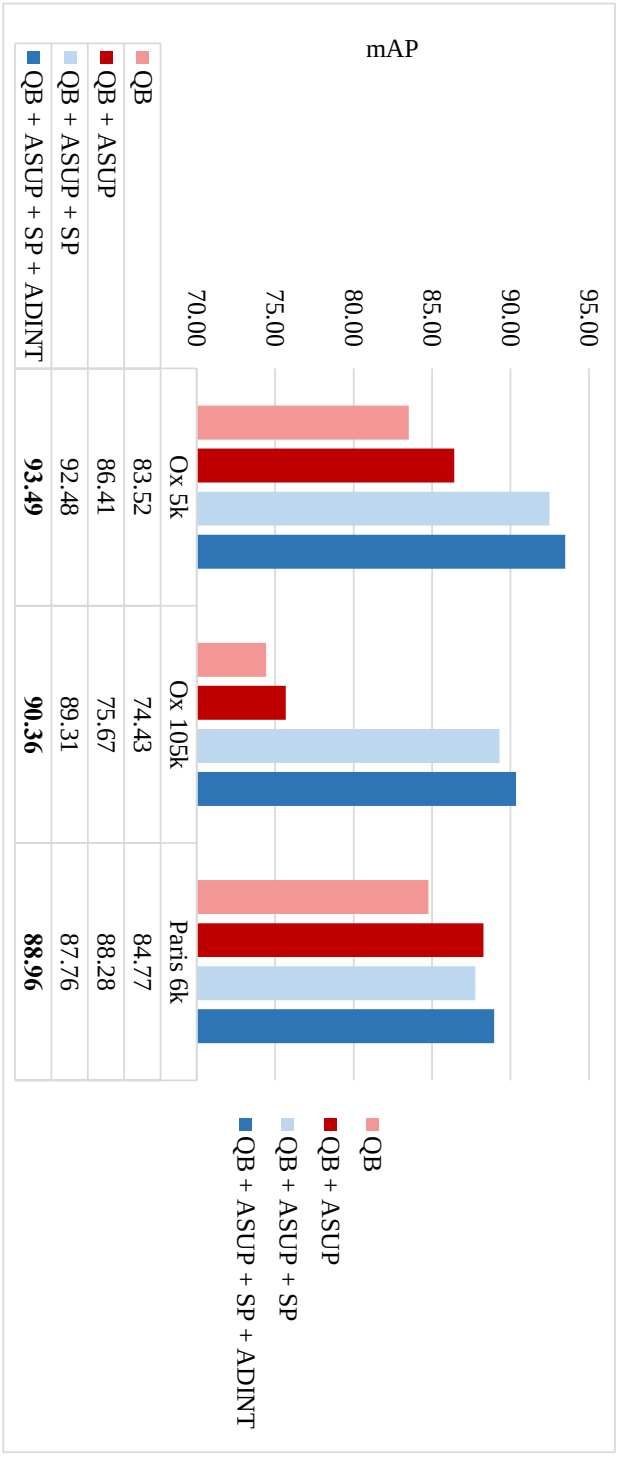
At this point, as we proposed several methods aimed to improve the final retrieval performance. Thereby, in this section, we are evaluating our presented methods by module to see how each module improves the result relatively to each others. There are 2 core modules and 2 additional parameter tuning algorithms as follows:

- Query bootstrapping (QB)
- Automatic support (ASUP)
- Spatial verification (SP)
- Adaptive inlier threshold (ADINT)

However, ASUP is a direct extension to QB, and ADINT is a direct extension to SP, and also SP is extended on QB. We then evaluate the performance based on with and without extension as follows:

- QB
- QB + ASUP
- QB + ASUP + SP
- QB + ASUP + SP + ADINT

The total result of component based evaluation is reported in the figure 6.4. We run the experiments on only the standard datasets of Oxford 5k, 105k and Paris 6k.



	Ox 5k	Ox 105k	Paris 6k
QB	83.52	74.43	84.77
QB + ASUP	86.41	75.67	88.28
QB + ASUP + SP	92.48	89.31	87.76
QB + ASUP + SP + ADINT	**93.49**	**90.36**	**88.96**

FIGURE 6.4: The performance comparison of our contributions shows the accumulated performance gain by adding each component to the standard BoVW baseline.

6.6 An impact of query quality to retrieval robustness

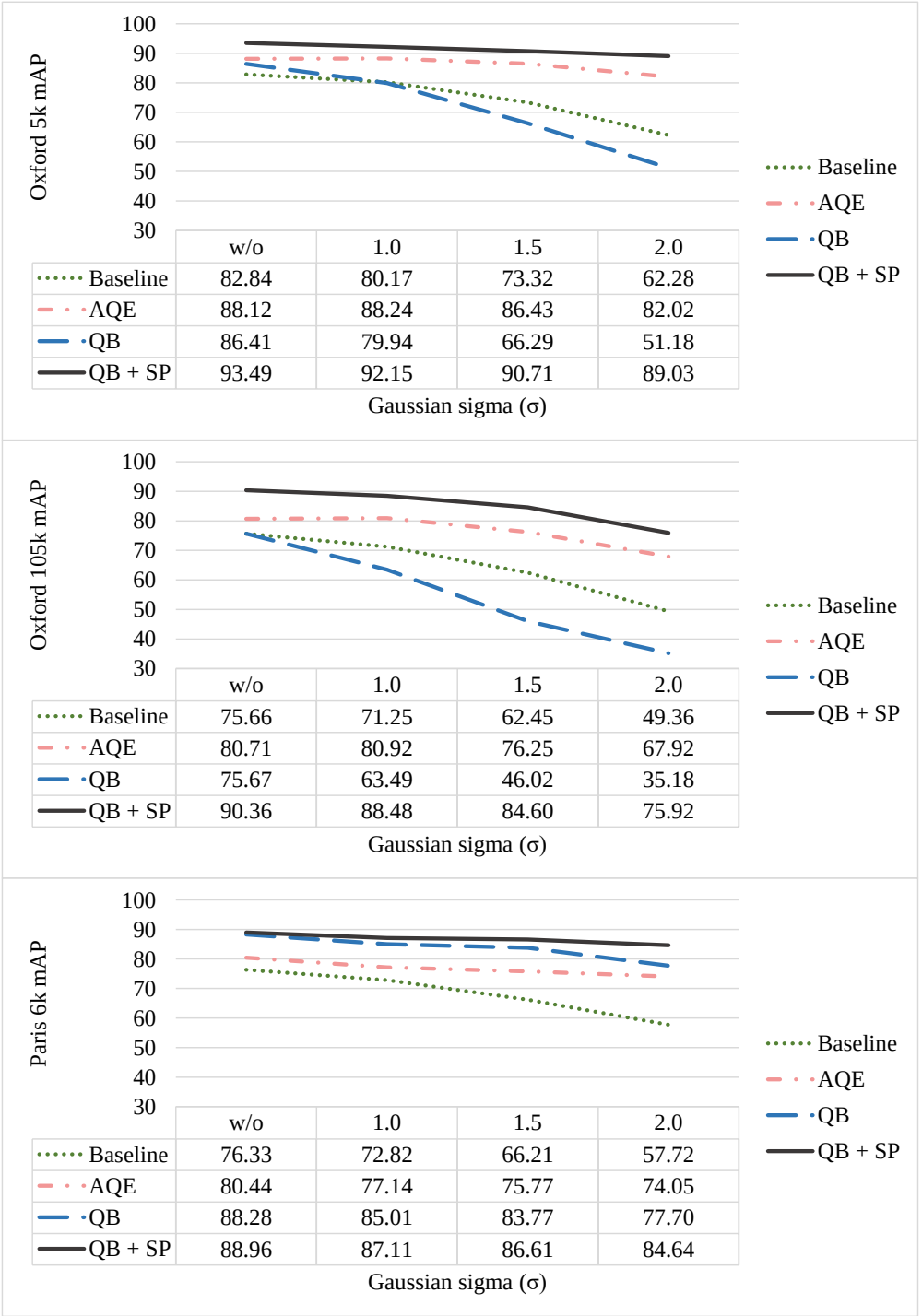
The following experiments assessed the impact of degraded queries.

6.6.1 Query with noise

We added the Gaussian noise with different standard deviation values (σ = 1.0, 1.5, and 2.0) to the original query. The results, reported in figure 6.5, show that QB + SP gave the best results for all three datasets and showed good robustness to noise levels. However, QB became inaccurate when very hard noise appeared in an image and put many more irrelevant images in the initial ranked lists. AQE and BoVW were affected in the similar way.

6.6.2 Query with lower resolution

We did another experiment on query quality that assumed the query image was from a low-resolution camera or was a thumbnail image that would be used, for example, to speed up data transfers through a high latency network. We resized the query on different scales (20%, 40%, 60%, and 80%). Figure 6.5 shows that QB + SP performed the best until 40% whereas other methods experienced large performance drops.

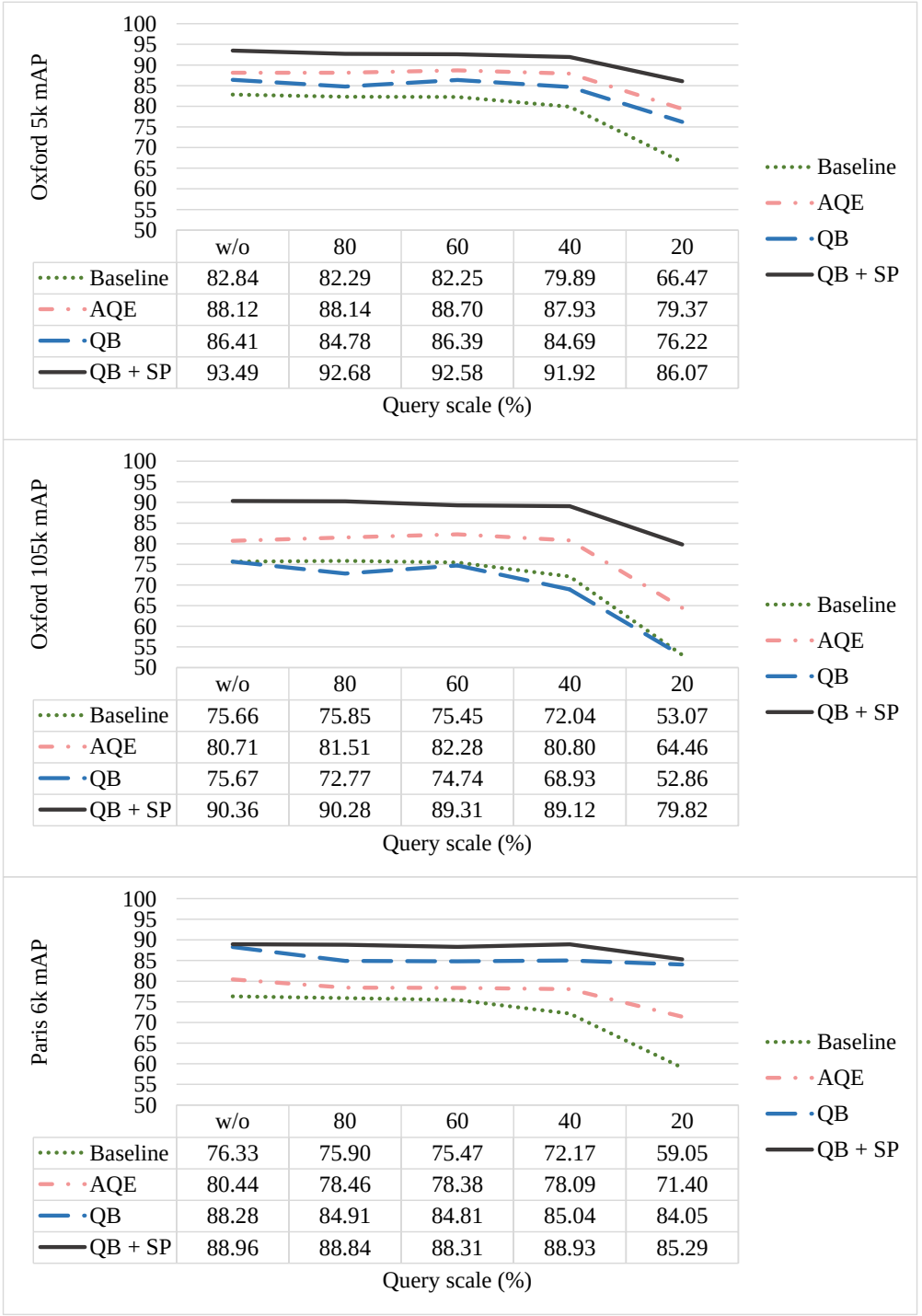


	w/o	1.0	1.5	2.0
Baseline	82.84	80.17	73.32	62.28
AQE	88.12	88.24	86.43	82.02
QB	86.41	79.94	66.29	51.18
QB + SP	93.49	92.15	90.71	89.03

	w/o	1.0	1.5	2.0
Baseline	75.66	71.25	62.45	49.36
AQE	80.71	80.92	76.25	67.92
QB	75.67	63.49	46.02	35.18
QB + SP	90.36	88.48	84.60	75.92

	w/o	1.0	1.5	2.0
Baseline	76.33	72.82	66.21	57.72
AQE	80.44	77.14	75.77	74.05
QB	88.28	85.01	83.77	77.70
QB + SP	88.96	87.11	86.61	84.64

FIGURE 6.5: Retrieval performance for synthetic noisy query.



	w/o	80	60	40	20
Baseline	82.84	82.29	82.25	79.89	66.47
AQE	88.12	88.14	88.70	87.93	79.37
QB	86.41	84.78	86.39	84.69	76.22
QB + SP	93.49	92.68	92.58	91.92	86.07

	w/o	80	60	40	20
Baseline	75.66	75.85	75.45	72.04	53.07
AQE	80.71	81.51	82.28	80.80	64.46
QB	75.67	72.77	74.74	68.93	52.86
QB + SP	90.36	90.28	89.31	89.12	79.82

	w/o	80	60	40	20
Baseline	76.33	75.90	75.47	72.17	59.05
AQE	80.44	78.46	78.38	78.09	71.40
QB	88.28	84.91	84.81	85.04	84.05
QB + SP	88.96	88.84	88.31	88.93	85.29

FIGURE 6.6: Retrieval performance for simulated low-resolution query.

6.7 Time consumption

We ran these experiments using the parameters that achieved the best performance for each of the methods. We used the top-100 images for AQE, QB + SP on all datasets, the top-100 for QB on the Paris dataset and the top-25 for QB on the Oxford datasets. Figure 6.7 lists the average time consumption per query. Note that we excluded feature extraction from the evaluation since it had no effect on any method.

QB + SP was the slowest, followed by QB, AQE, and BoVW. We did a time breakdown on QB and QB + SP, namely, on the SP (spatial verification) module, FIMT (frequent itemset mining process with matrix transposition) module, and Sim (similarity calculation) module. It is clear from figure 6.8 that the most time-consuming module was FIM process. On the other hand, FIM tends to be slow when the number of discovered patterns is huge. Assuming that if a query is "easy" in terms of retrieval performance (namely, BoVW can achieve very high performance and thus the contribution of query expansion is limited), many visual words may be found in common among the query and relevant images, and thus, FIM may produce many patterns. Moreover, if we assume that a "hard" query (BoVW performs poorly and thus there is room for the improvement by query expansion), relevant images may have fewer visual words in common, FIM may produce fewer patterns. To see whether this assumption is valid, we checked the relationship between retrieval performance and the total number of patterns. Figure 6.9 plots mAP values for each query versus the number of patterns for all methods.

For a query on which BoVW obtained a low mAP, FIM discovers fewer patterns and thus is fast. On the other hand, for a query on which BoVW obtained a high mAP, FIM tends to produce more patterns. Accordingly, we categorized queries into two types using the number of patterns, which are *easy* for FIM (#patterns $<$ 100k) and *hard* for FIM (#patterns $\geq$ 100k).

The red vertical lines in figure 6.9 show boundaries between easy and hard cases. On the easy side, QB + SP significantly improved mAP. However, it yielded less of an improvement on the hard side. Table 6.2 shows mAP values and processing times for easy and hard cases. QB + SP consumed much more processing time for the hard cases than for the easy cases, whereas its performance improvement was significantly larger in the easy cases than in the hard cases.

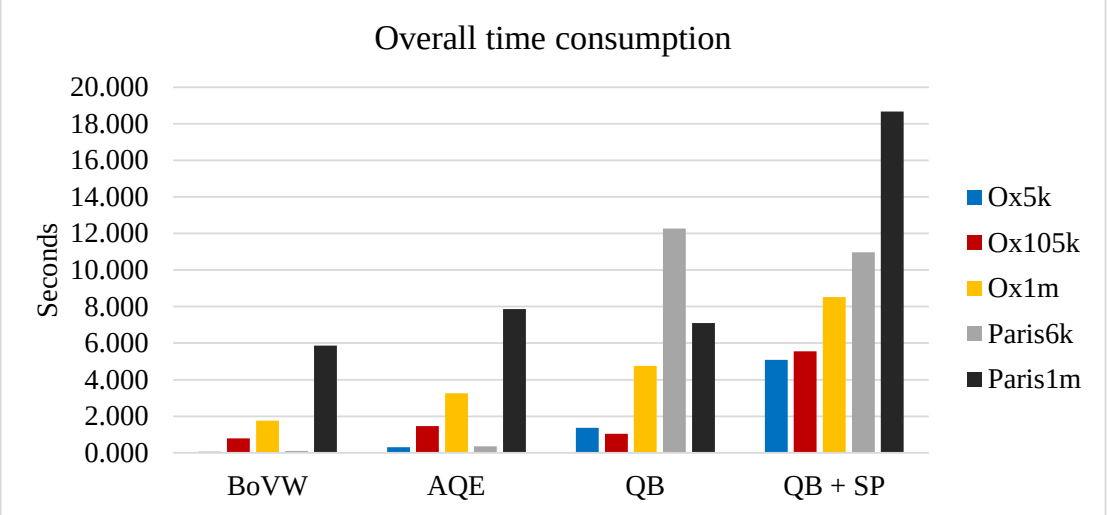


FIGURE 6.7: Average time consumption per query (without feature extraction) of BoVW, AQE, QB, and QB + SP.

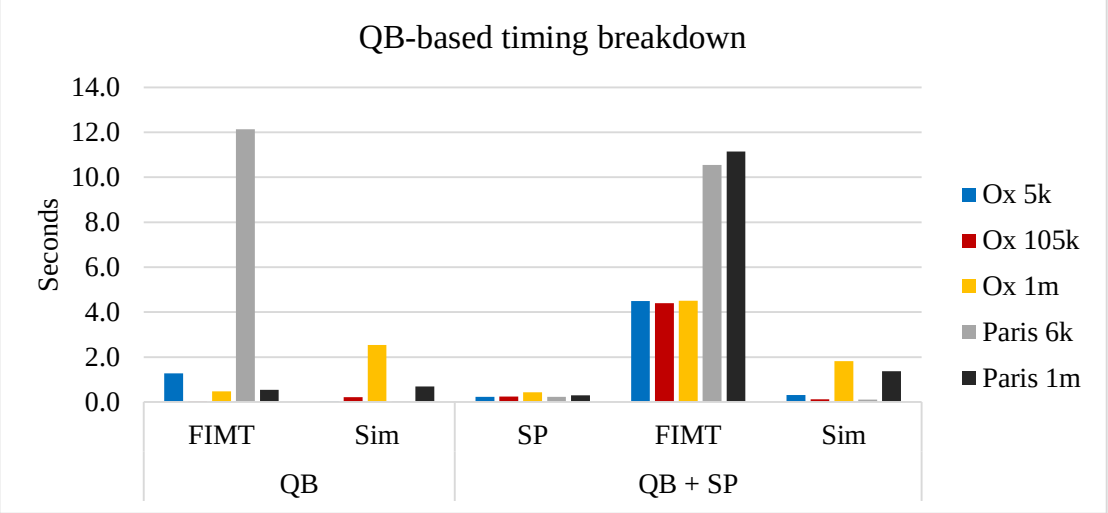


FIGURE 6.8: Average time consumed per query for each module of QB and QB + SP.

6.7.1 Colossal pattern

Although we used transaction transposition to speed up the pattern discovery process, we still spent a lot of time on the "hard" cases. Overall, these queries made QB + SP significantly slower on average than the other methods, as shown in table 6.2.

The main reason behind this slowdown is that the large number of relevant images with many visual words in common caused an explosion in the number of patterns discovered by FIM. This problem is called the colossal pattern problem. In [35], it is pointed out that not all of these patterns may need to be discovered and their patterns can be roughly estimated by using an approximate method [133]. In our study, while mining "easy" queries could be accomplished very quickly (within a few hundred milliseconds), the

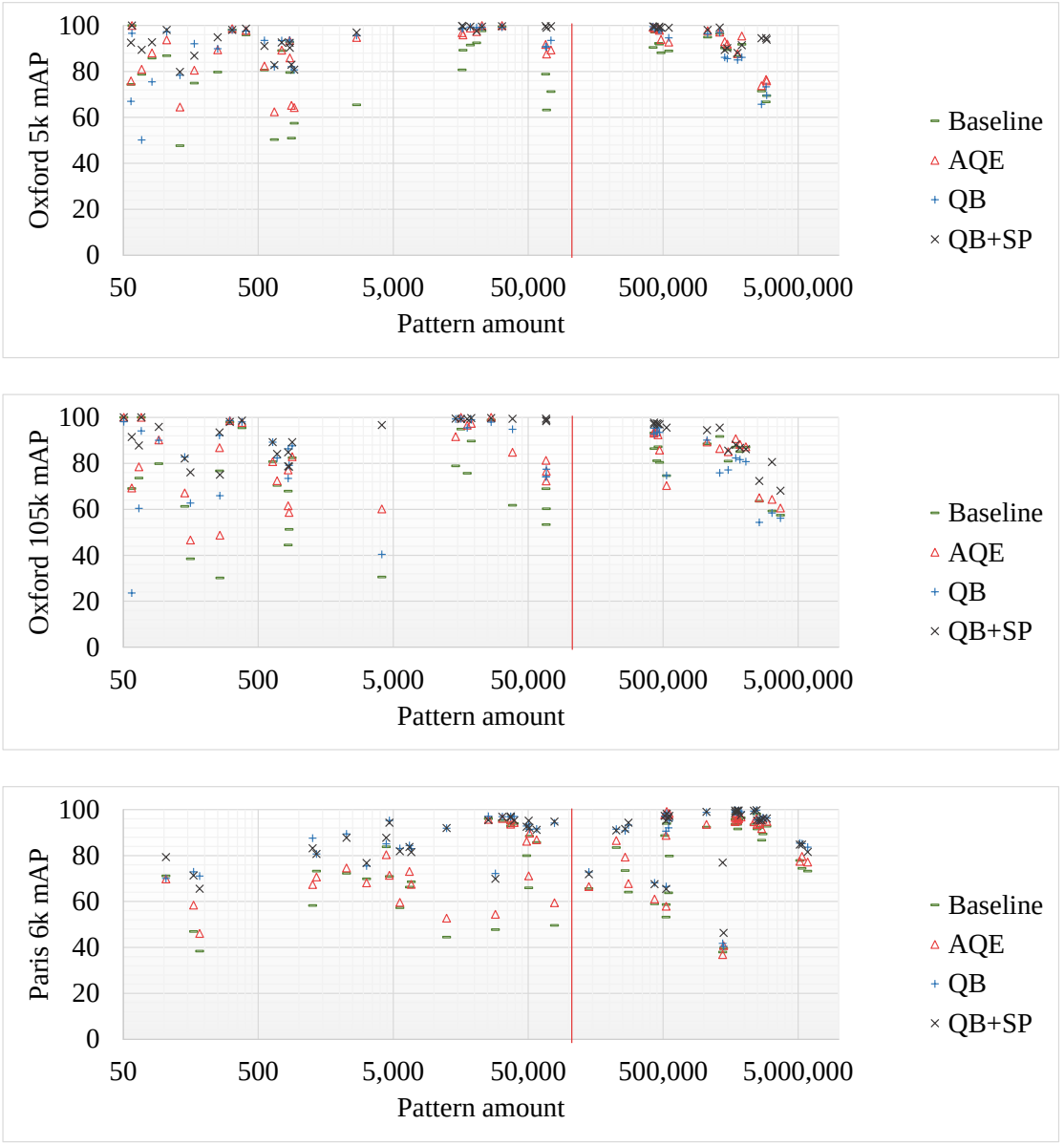


FIGURE 6.9: mAP vs. total number of patterns. This plot shows the improvement was had by our QB + SP on most queries that generated less than 100k patterns. In contrast, it had less effect on queries that generated more than 100k patterns.

"hard" cases were time consuming. According to the experimental results in figure 6.10, mAP improved only slightly for the total number of patterns over 100k. Here, we could stop mining the "hard" cases once FIM has discovered enough of the core patterns, which took less than 3 seconds to compute. Accordingly, thanks to transaction transposition technique, we can recover the most of approximated patterns from the colossal space (e.g., 10 million patterns).

	Type	#Topics	BoVW	QB				QB+SP			
				FIM^T(s)	Precision(%)			FIM^T(s)	Precision(%)		
			mAP(%)		mAP(%)	SD(±%)	mAP+(%)		mAP(%)	SD(±%)	mAP+(%)
Ox 5k	Easy	40	81.26	0.075	85.51	21.02	4.25	0.166	92.69	14.25	**11.43**
	Hard	15	87.06	4.471	88.79	10.97	1.72	16.037	95.64	4.07	**8.58**
Ox 105k	Easy	40	73.94	0.011	73.99	29.94	0.05	0.066	90.77	15.95	**16.83**
	Hard	15	80.24	0.109	80.13	13.81	-0.11	15.949	89.28	9.19	**9.04**
Ox 1m	Easy	40	72.92	0.045	75.51	29.76	2.59	0.131	85.42	19.98	**12.50**
	Hard	15	81.58	1.601	83.04	12.48	1.46	16.184	90.45	7.29	**8.87**
Paris 6k	Easy	25	71.09	0.922	86.53	9.23	**15.44**	0.363	86.17	9.39	15.08
	Hard	30	80.69	21.475	89.74	15.37	9.05	19.030	91.28	12.28	**10.59**
Paris 1m	Easy	25	49.23	0.066	63.58	23.01	**14.35**	0.281	70.89	21.63	21.65
	Hard	30	68.88	0.927	75.23	19.87	6.34	20.196	79.76	19.78	**10.87**

TABLE 6.2: mAP and times of specific query types (easy/hard). FIM takes the longest and is especially slow on "hard" queries that mostly got high retrieval performance in the first round. QB + SP yielded a very large performance improvement with very low latency for "easy" queries.

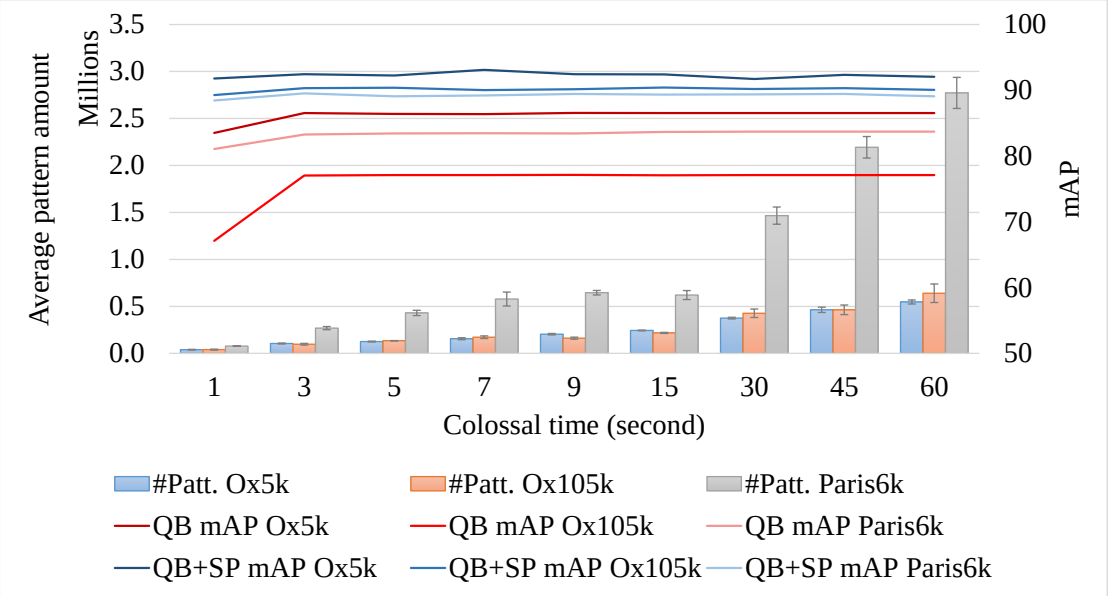


FIGURE 6.10: Number of patterns versus time. The discovery of rest patterns yielded no big improvement in mAP after it had produced enough core patterns about 100k patterns, which were processed within 3 seconds.

6.8 Retrieval result examples and analysis

In this section, we select the example of the retrieved images using all previously evaluated methods e.g., BoVW, AQE, QB, and QBSP to show how our proposed methods are good in the different cases e.g., a normal query, a small object, a low resolution, and also a noisy case.

6.8.1 Normal query case

We selected the sample of this normal query case from the result of Oxford 1M dataset, on the topic of "christ_church_5". As result (figure 6.11), standard BoVW provide the first round retrieved images with AP of 41.60%, AQE has an AP of 80.82%, our QB has an AP of 88.84%, and the best one is QB + SP with an AP of 96.58% respectively.

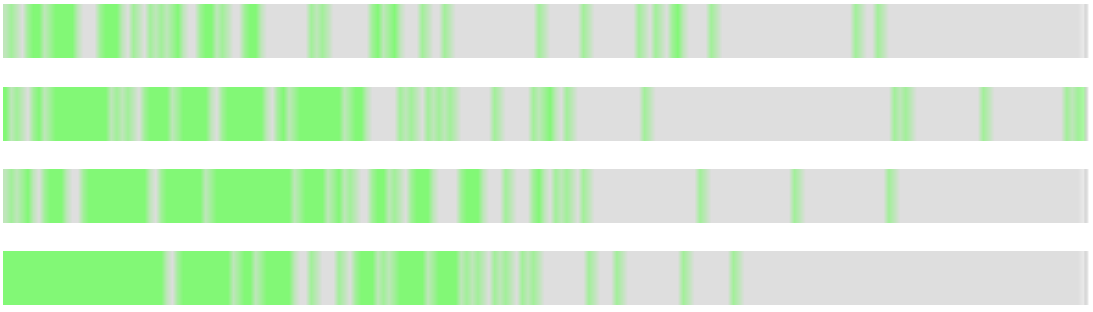


FIGURE 6.11: The illustration of true-positive list on a retrieved images using a normal query with BoVW, AQE, QB, and QB + SP

Normal query case: BoVW

FIGURE 6.12: BoVW Matching result with a normal query. AP = 40.16%

Normal query case: AQE

FIGURE 6.13: AQE Matching result with a normal query. AP = 80.82%

Normal query case: QB

FIGURE 6.14: QB Matching result with a normal query. AP = 88.84%

Normal query case: QBSP

FIGURE 6.15: QB + SP Matching result with a normal query. AP = 96.58%

6.8.2 Small object query case

We selected the sample of this small object query case from the result of Oxford 105k dataset, on the topic of "magdalen_2". As result (figure 6.16), standard BoVW provide the first round retrieved images with AP of 57.47%, AQE has an AP of 66.51%, our QB has an AP of 80.75%, and the best one is QB + SP with an AP of 81.68% respectively.

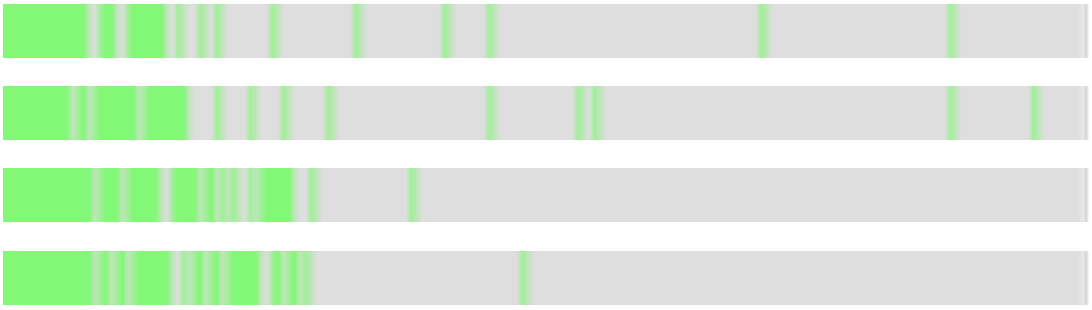


FIGURE 6.16: The illustration of true-positive list on a retrieved images using a small object query image with BoVW, AQE, QB, and QB + SP

Small object query case: BoVW

FIGURE 6.17: BoVW Matching result with a small object query. AP = 57.47%

Small object query case: AQE

FIGURE 6.18: AQE Matching result with a small object query. AP = 66.51%

Small object query case: QB

FIGURE 6.19: QB Matching result with a small object query. AP = 80.75%

Small object query case: QBSP

FIGURE 6.20: QB + SP Matching result with a small object query. AP = 81.68%

6.8.3 Low resolution query case

We selected the sample of this low resolution query case (20% scaled of original) from the result of Paris 1M dataset, on the topic of "moulinrouge_2". As result (figure 6.21), standard BoVW provide the first round retrieved images with AP of 45.59%, AQE has an AP of 45.59%, our QB has an AP of 91.20%, and the best one is QB + SP with an AP of 92.30% respectively.

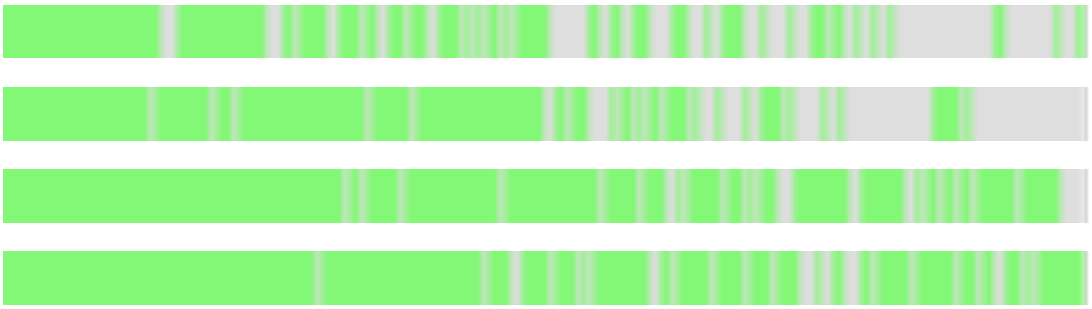


FIGURE 6.21: The illustration of true-positive list on a retrieved images using a low resolution query image with BoVW, AQE, QB, and QB + SP

Low resolution query case: BoVW

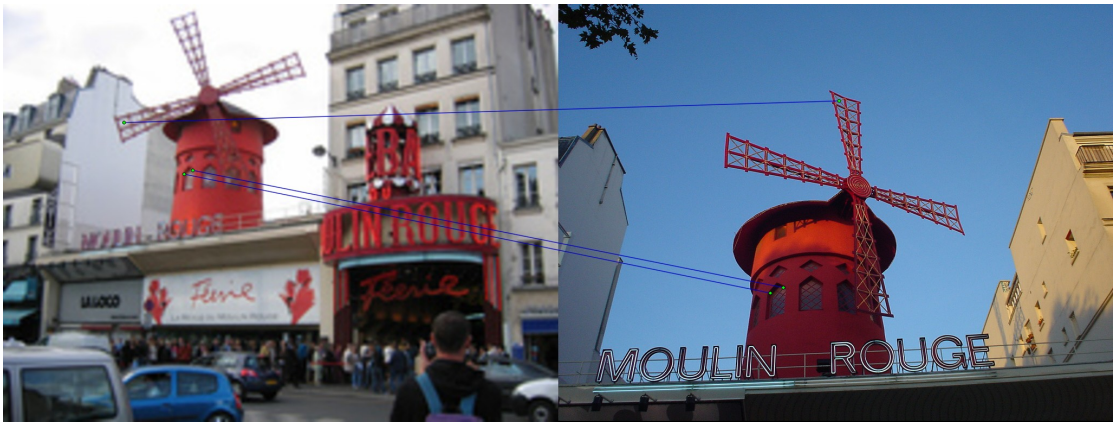


FIGURE 6.22: BoVW Matching result with a 20% scale of query. AP = 45.59%

Low resolution query case: AQE

FIGURE 6.23: AQE Matching result with a 20% scale of query. AP = 59.48%

Low resolution query case: QB

FIGURE 6.24: QB Matching result with a 20% scale of query. AP = 91.2%

Low resolution query case: QBSP

FIGURE 6.25: QB + SP Matching result with a 20% scale of query. AP = 92.3%

6.8.4 Noisy query case

We selected the sample of this noisy query case from the result of Paris 6k dataset, on the topic of "defense2". As result (figure 6.26), standard BoVW provide the first round retrieved images with AP of 8.20%, AQE has an AP of 37.47%, our QB has an AP of 1.74%, and the best one is QB + SP with an AP of 72.52% respectively.

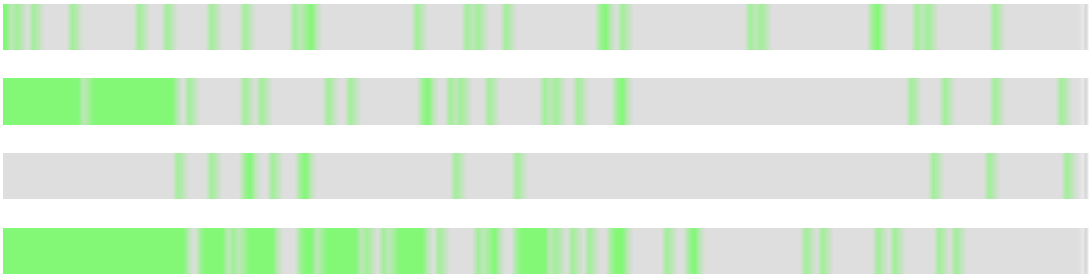


FIGURE 6.26: The illustration of true-positive list on a retrieved images using a noisy query image with BoVW, AQE, QB, and QB + SP

Low resolution query case: BoVW

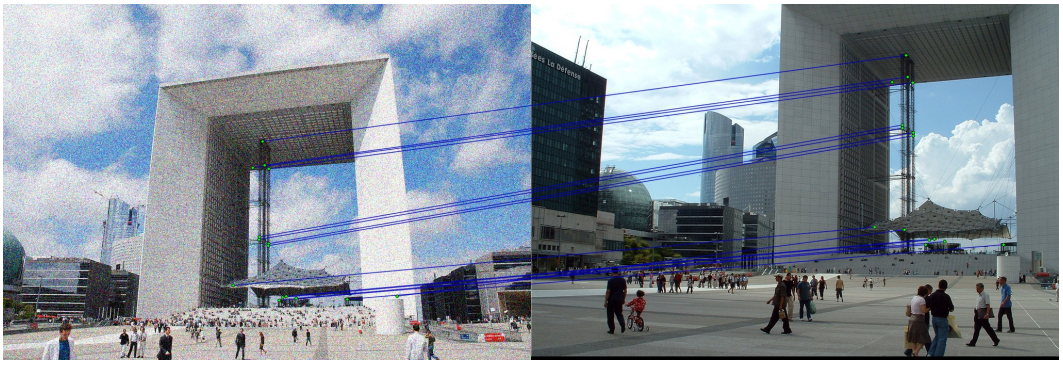


FIGURE 6.27: BoVW Matching result with a noisy query. AP = 8.20%

Low resolution query case: AQE

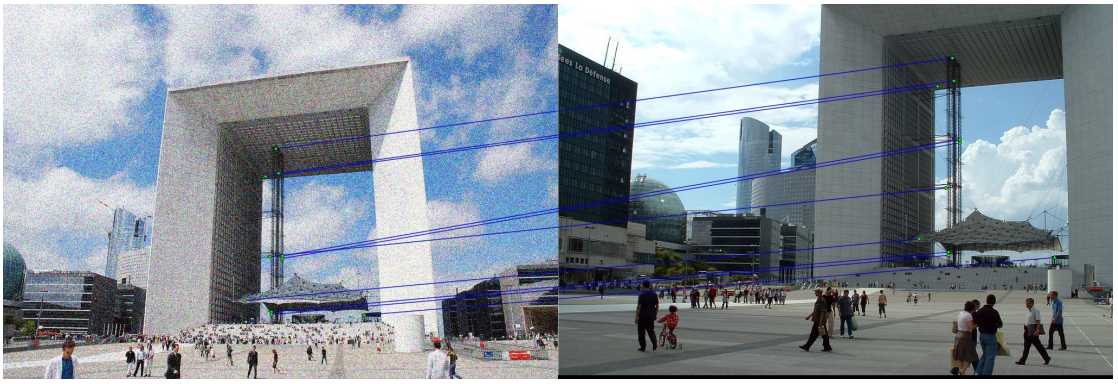


FIGURE 6.28: AQE Matching result with a noisy query. AP = 37.47%

Low resolution query case: QB (failed)

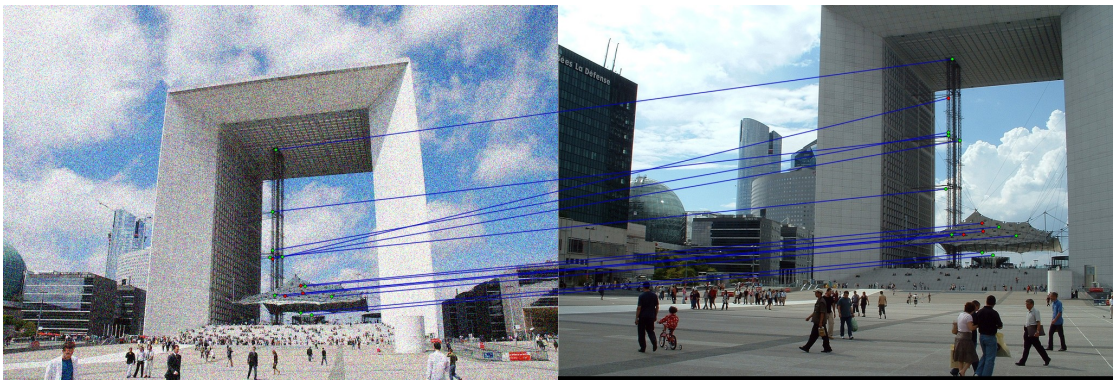


FIGURE 6.29: QB Matching result with a noisy query. AP = 1.74%

Low resolution query case: QBSP

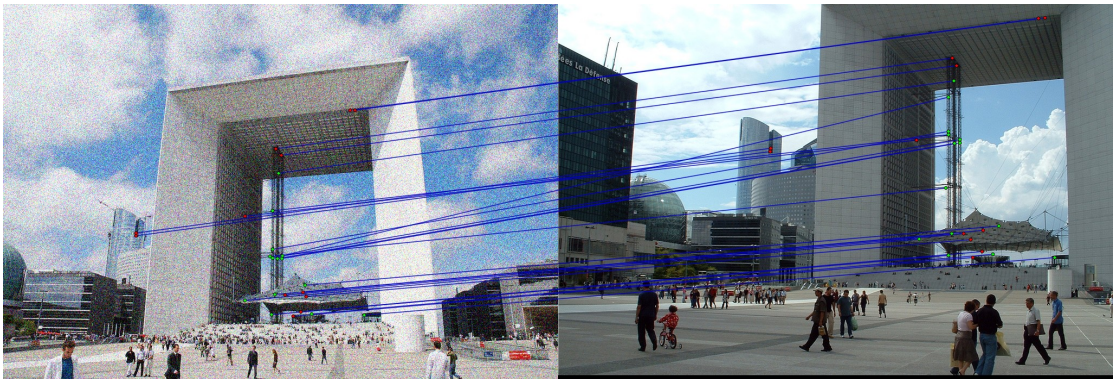


FIGURE 6.30: QB + SP Matching result with a noisy query. AP = 72.52%

6.9 Discussion

We presented QB which aimed for relaxing the spatial verification constraint from a degraded query image. Indeed, QB has several benefits that directly derived from FIM tools as FIM can be used to discover the existing of the co-occurring of visual word groups that forming a visual object. Together with QB also help in understanding more on the target object as the target object can be learn by using multiple image together. In the same time, using QB need to target on the datasets with some criteria, at least, the first round result has to be good enough at some level. So, in this section, we will discuss in details about the benefits, weakness, and the limitation of using our QB.

6.9.1 Benefits of using QB

As we employed FIM for finding patterns that co-occurring throughout of multiple relevant images, the major benefit of FIM is that the discovered patterns will be corresponded to the object and has less ambiguity meaning of the overall patterns regardless of the specific foreground and background region. So, by using multiple images, QB will help understand more on the target object and its context. In this sense, the context surrounding an object can be learned, and also the hidden visual words from the other view angles can also be learned. Moreover, since our methods can help in learning about the object and its context, rejecting the irrelevant visual words will also be easy. The problem that QB can handle or reject quite well are, object occlusions, misleading visual words, not useful visual words, and the visual words that not much clearly related to the target object.

6.9.1.1 Context discovery

QB can help learning on the context for a specific target object. The first selected example, a query is given as in the figure 6.31, as to search for the object within the specific ROI.

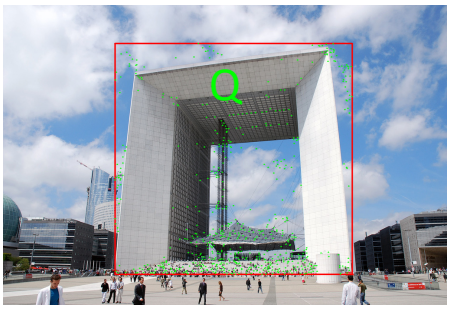


FIGURE 6.31: A query example with the feature point within a specified ROI.

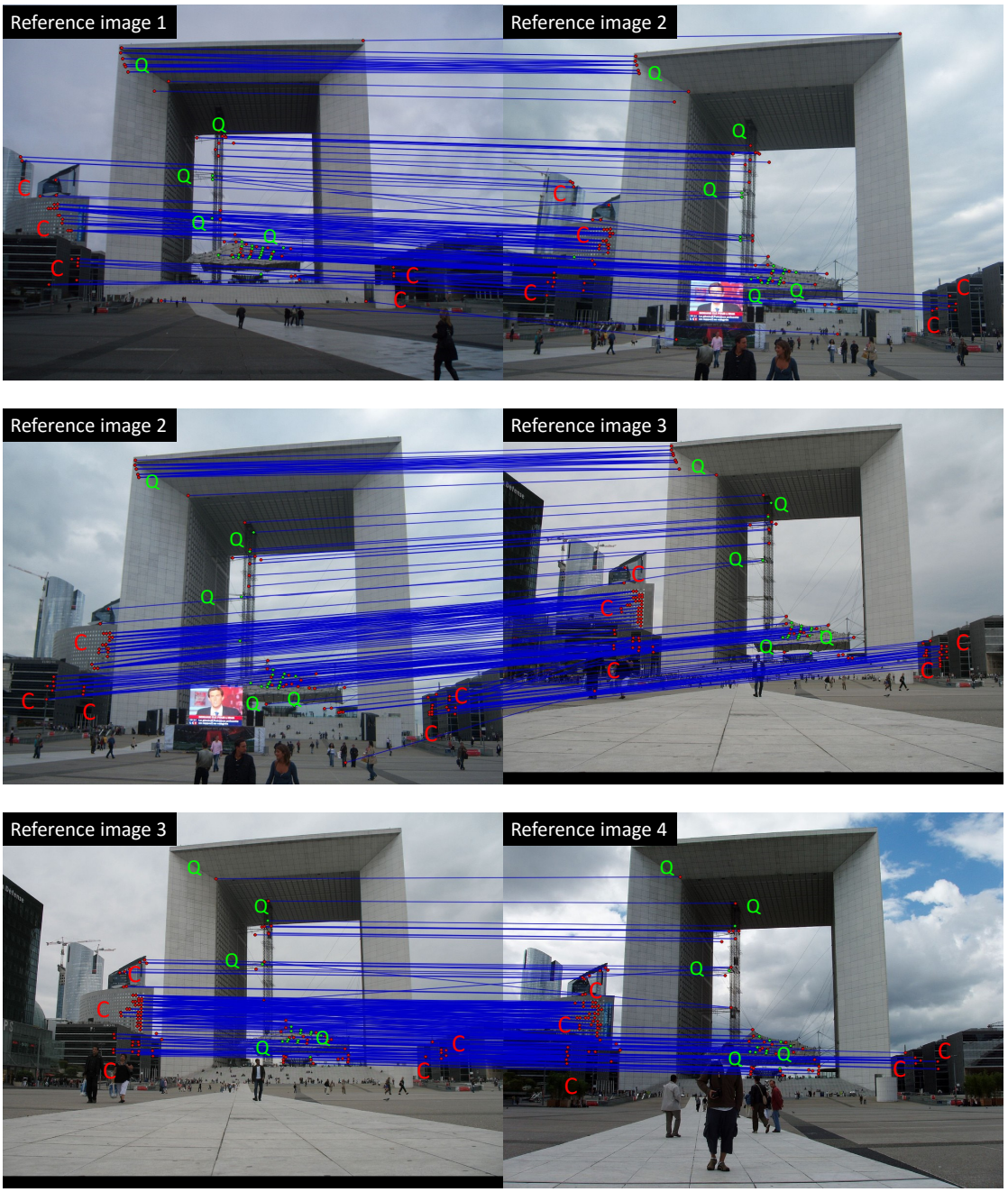


FIGURE 6.32: The query example for QB context discovery shows the matching of query 'Q's are belonging to the target object, while the matching of context 'C's are belonging to the context information surrounding the target object.

The QB task is to find the co-occurrences of patterns between the top relevant images regardless the foreground and background region. By doing this, each of the images will be discovered for the patterns both inside and outside of the target ROI as shown as 'Q' (matching to the target object) and 'C' (matching to the surrounding context) in the figure 6.32.

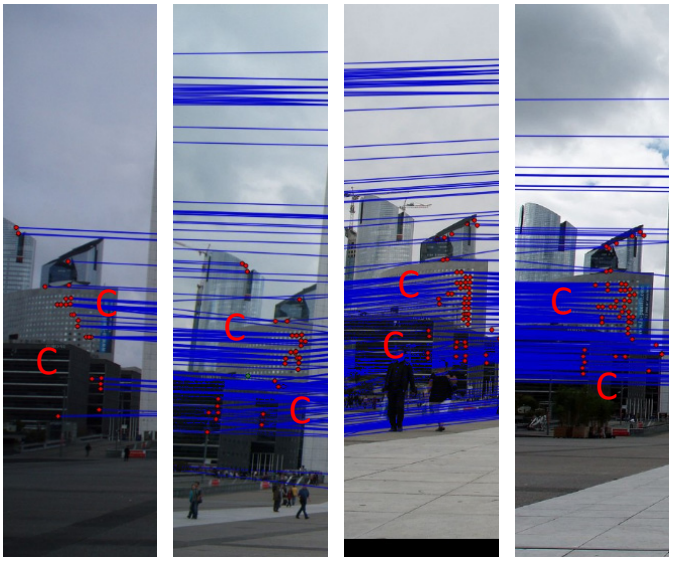
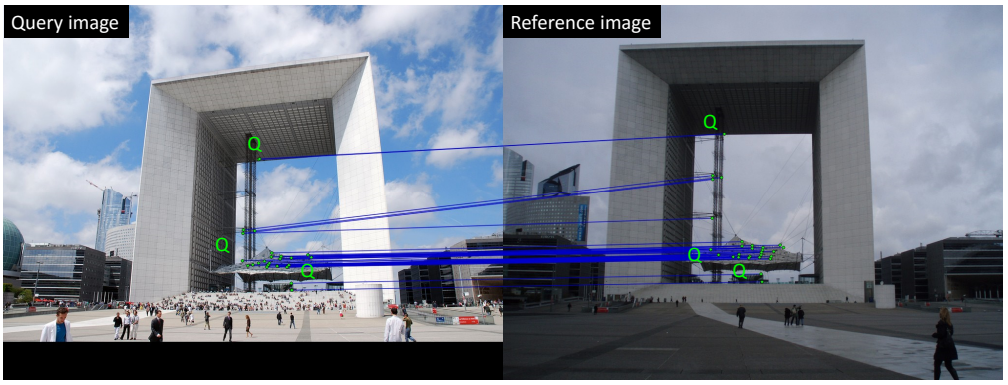
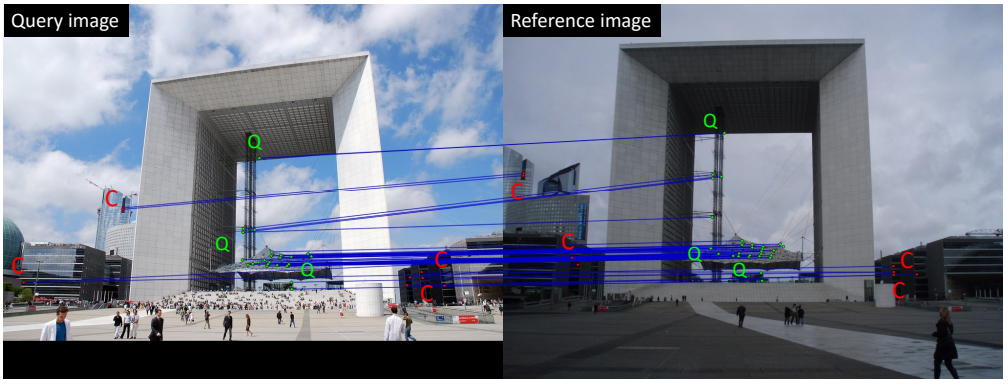


FIGURE 6.33: The contexts discovered from top 4 relevant images by using QB. As 'C's represent the context such as the surrounding building is always beside the target building, therefore, the beside building can be used in retrieving the target building correctly.



(A)



(B)

FIGURE 6.34: The matching result shows how the selected example matches to the query image for a topic "defense_2' on a Paris 1M dataset. (A) The matching by using AQE will calculating the similarity within only the specified ROI, in this example, AQE obtain an AP of 27.40%. (B) The matching by using our QBSP will utilize the context information to help in finding the target query, in this example, our QBSP obtain an AP of 71.35%.

Therefore, our QB not just only discover the patterns within the target object, but also it can find the context around (see figure 6.33) to help explain on the object itself. As result, using context can help improve the final retrieval performance as shown in the figure 6.34. Another example of context discovery is as the following selected query in figure 6.35. Also, we show the context (red ellipses) can be found in this second example in the figure 6.36.

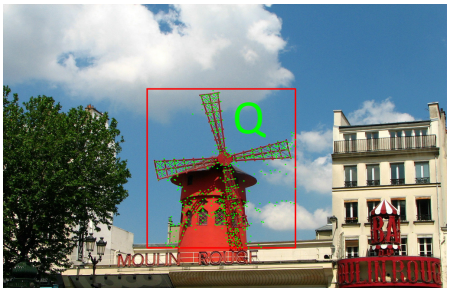


FIGURE 6.35: A query example with the feature point within a specified ROI.

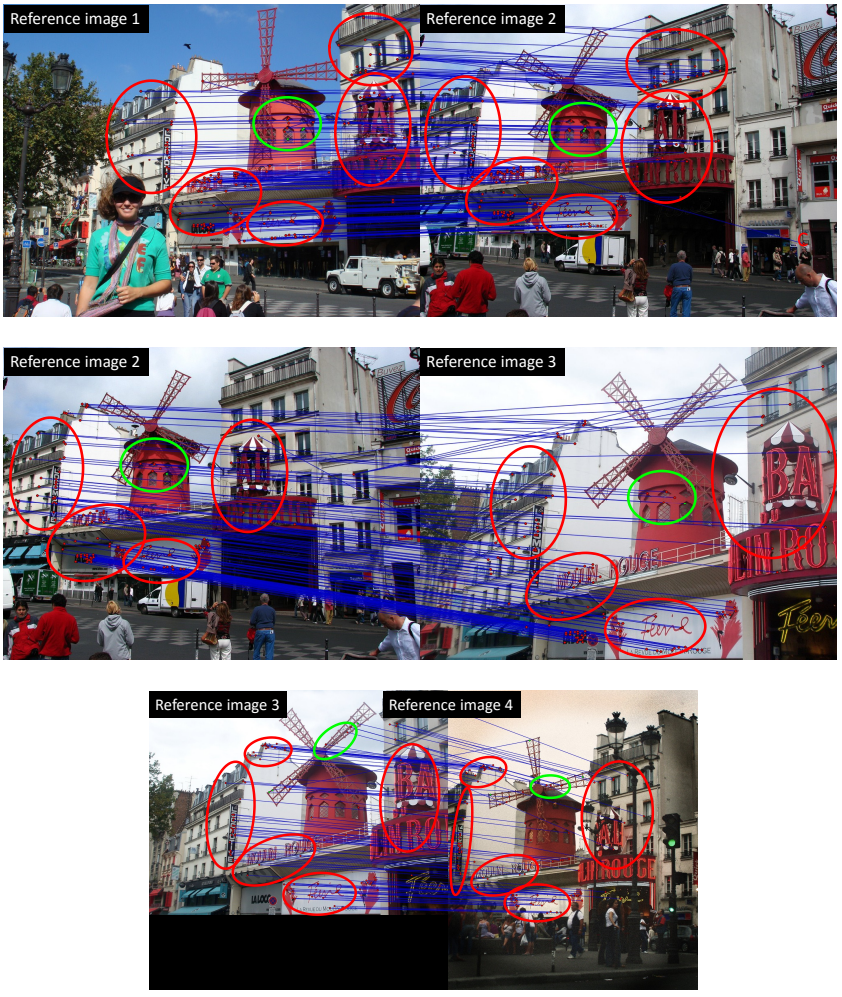
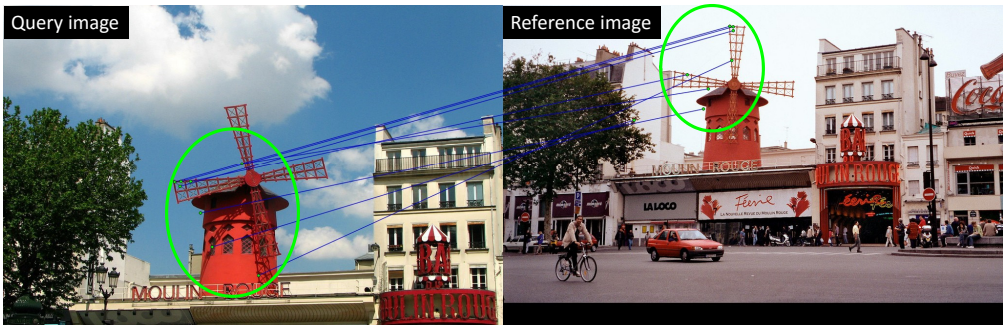


FIGURE 6.36: The query example for QB context discovery shows the matching of query 'green ellipse's are belonging to the target object, while the matching of context 'red ellipse's are belonging to the context information surrounding the target object.



(A)

(B)

FIGURE 6.37: The matching result shows how the selected example matches to the query image for a topic "defense_2' on a Paris 1M dataset. (A) The matching by using AQE will calculating the similarity within only the specified ROI, in this example, AQE obtain an AP of 28.86%. (B) The matching by using our QBSP will utilize the context information to help in finding the target query, in this example, our QBSP obtain an AP of 83.52%.

As result, using context can help improve the final retrieval performance as shown in the figure 6.37.

6.9.1.2 Hidden visual words discovery

In some cases, a given query image does not provided in clear and low detail of the texture on a target object. Even generally an object is not just a planar, the detail on the other perspectives can help in learning the hidden feature by using multiple view angle. The example of this situation is shown in the figure 6.38. As we can see the matching result, the figure 6.39 shows that our QB can find the hidden visual words within the target query image, much higher than what AQE did. And the figure 6.40 shows the result of retrieving with hidden visual word discovery achieved much better performance (AQE with AP of 23.67%, QB with AP of 44.77%).

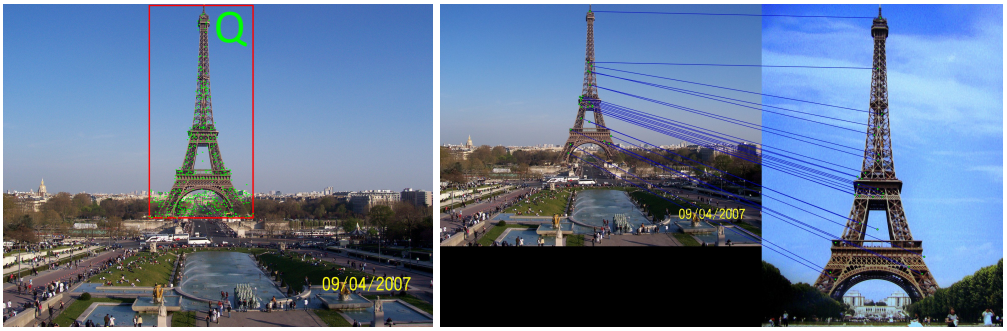


FIGURE 6.38: A query example with the feature point within a specified ROI.

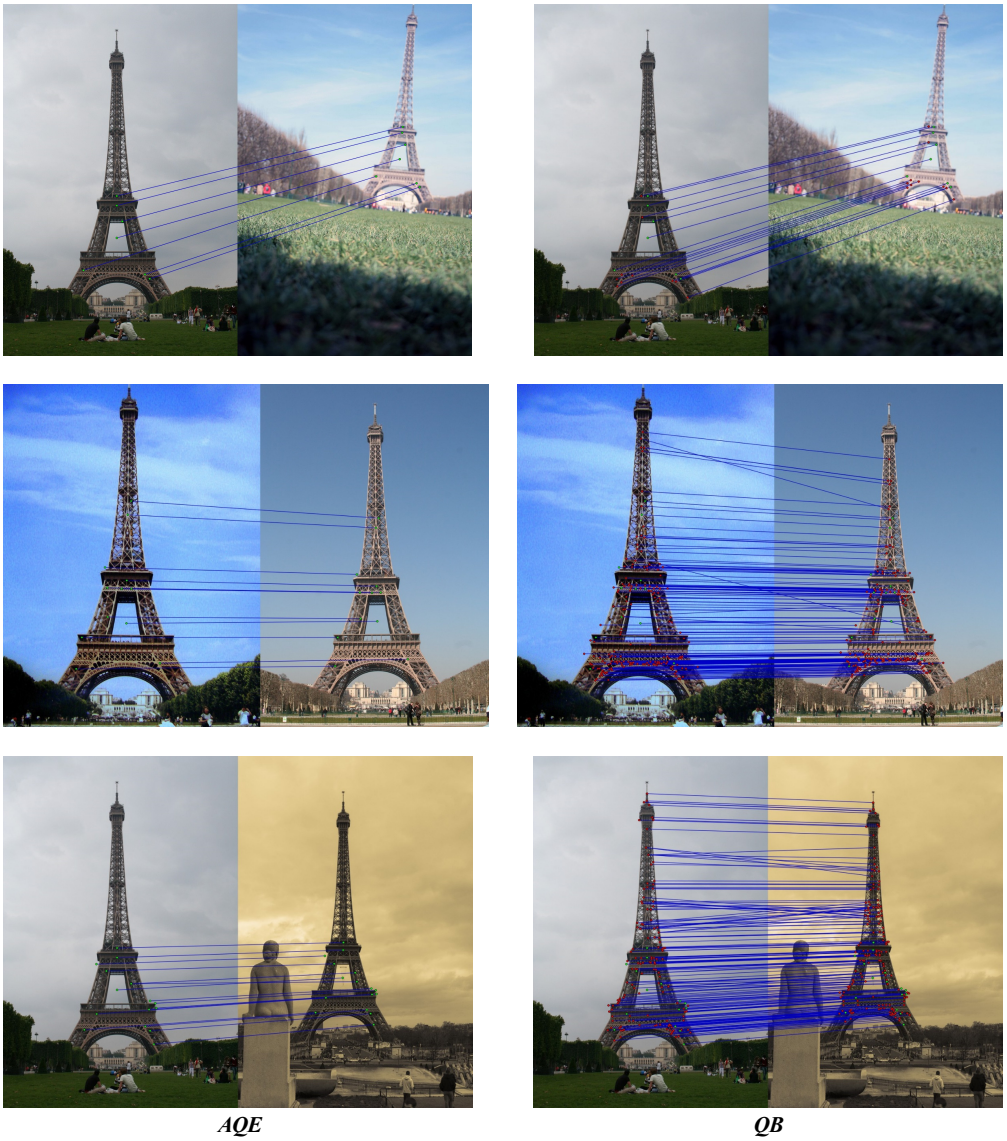


FIGURE 6.39: The query example for QB context discovery shows the matching of query 'green ellipse's are belonging to the target object, while the matching of context 'red ellipse's are belonging to the context information surrounding the target object.

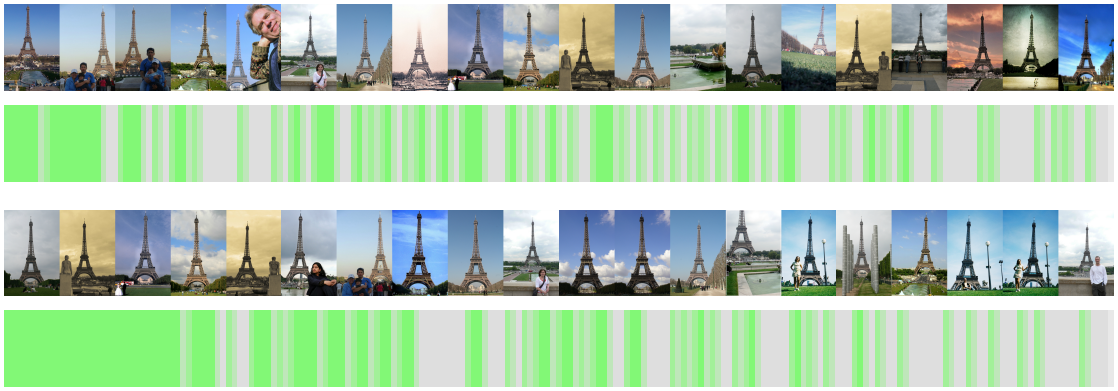


FIGURE 6.40: The selected results show the top 20 retrieved images from of the different method as by using AQE (top row), and our QB with the hidden visual words (bottom row). Top 100 result (true positive is in green) shows that our QB achieved better performance as AQE got an AP of 23.67% while our QB got an AP of 44.77%.

6.9.1.3 Reject irrelevant words

Also, our QB has better accurate in order to reject an irrelevant visual words. Since we employed FIM, QB then identify these problems wisely e.g., object occlusions, misleading visual words, and not clearly related to the target object. So, the following example shows the cases when our QB rejecting an irrelevant visual words perfectly.

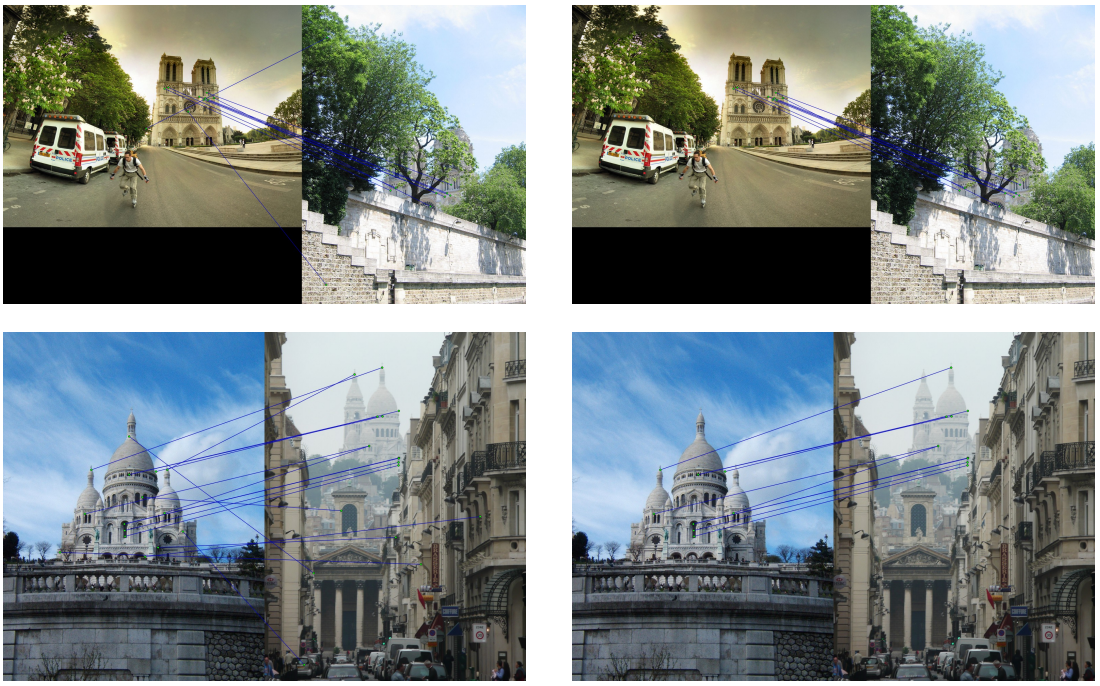


FIGURE 6.41: The example shows how AQE (left) and QB (right) reject irrelevant visual words in the different situation.

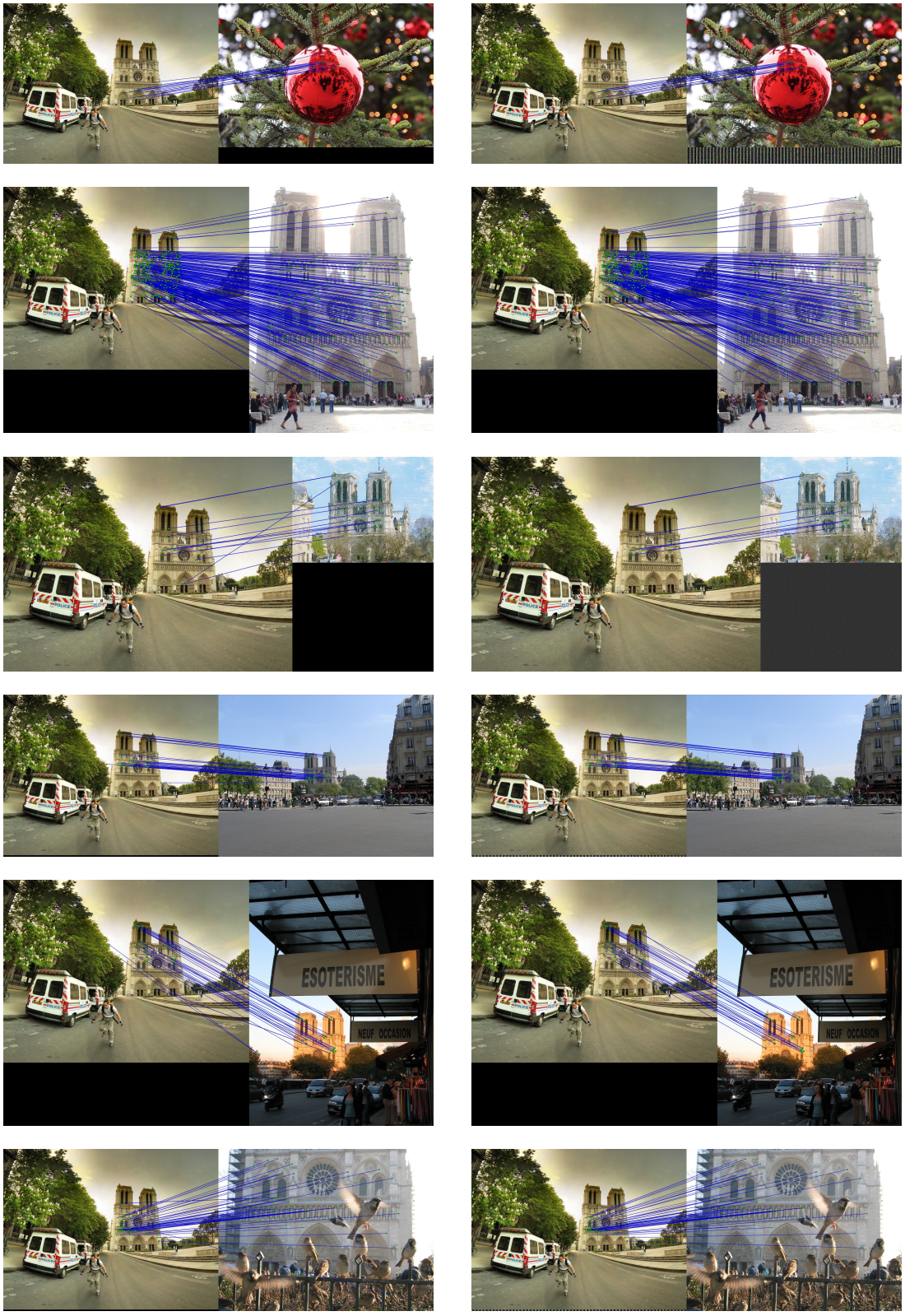


FIGURE 6.42: The example shows how AQE (left) and QB (right) reject irrelevant visual words in the different situation.

6.9.2 QB Limitations

In this section, we will discuss about the weakness and limitations of our QB-based approach. To reach the limitation, we tested our QB with the other datasets, as to test with the different target object perspective, database size, and moreover, as long as currently we conduct our former experiment with the image database, in this section, we also discuss how do we apply QB with the video database.

6.9.2.1 Experiments with the other datasets

For the previous experiments, we did the experiments based on Oxford and Paris dataset which are focused on retrieving the landmark images. In this section, we include the external 3 datasets, which are Stanford mobile visual search (MVS) [71] and the TrecVid instance search (INS) 2011-2013, for extending our experiments in term of the target objective and the type of the database.

Stanford mobile visual search dataset The dataset comes with 8 query topics, namely, book covers, business cards, cd covers, dvd covers, landmarks, museum paintings, prints, and video frames. The dataset has several key characteristics lacking in existing datasets: rigid objects, widely varying lighting conditions, perspective distortion, foreground and background clutter, realistic ground-truth reference data, and query data collected from heterogeneous low and high-end camera phones. However, since the dataset itself provide us only one true positive per query topic (see figure 6.43), the integration of QB is not possible as the FIM need multiple relevant images to find the patterns among the retrieved images.

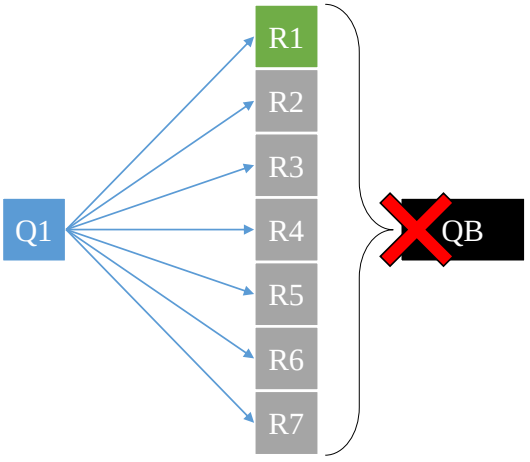


FIGURE 6.43: A trial illustration on integration of MVS dataset with our QB.

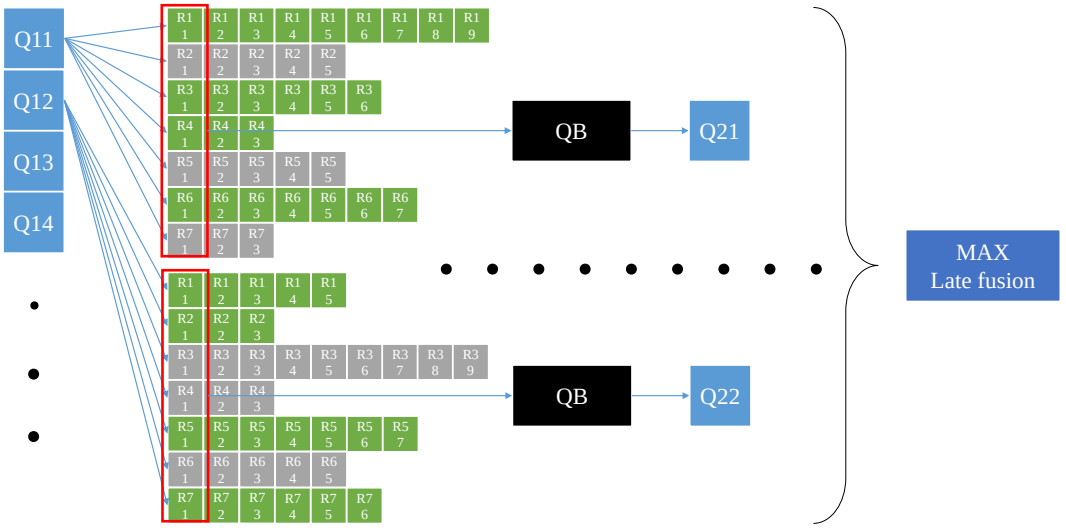


FIGURE 6.44: An illustration on how to integrate video frames from the INS datasets with our QB. The rectangle is covered on the first frame each shot result, aimed for being a representative frame during a spatial verification process of QBSP.

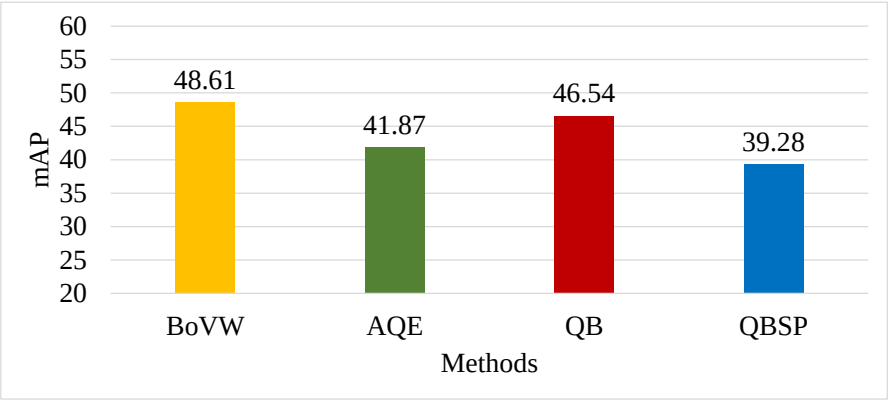
TrecVid instance search dataset The datasets are obtained from the a yearly workshop namely TRECVID instance search task [72]. There are 2 datasets which were integrated to be evaluated in our work, namely, instance search (INS), in particular, INS 2011 and INS 2013. The dataset are given as a collection of test clips (files) and a collection of queries that delimit a person, object, or place entity in some example video, locate for each query up to the 1,000 clips most likely to contain a recognizable instance of the entity. The objective of this retrieval is, a run return many fewer than 1,000 clips that related to the given query which will be evaluated by using mean average precision as an evaluation metric.

Since the datasets are based on video frame, here we will discuss a basic way to integrate our QB with such video frames. As illustrated in the figure 6.44, a query is provided with multiple frames per one query. Each sub query need to be searched separately and finally being fused by using max late fusion for scoring scheme. The integration to QB is as follow:

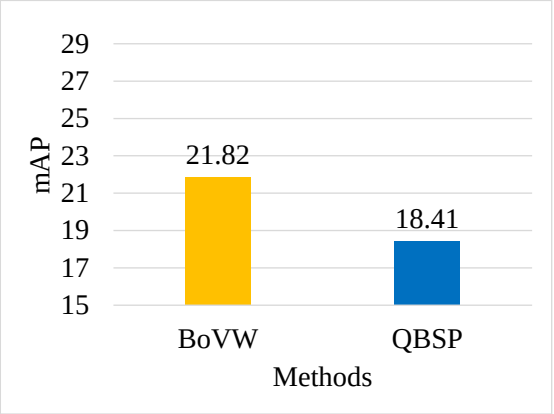
1. Each query need to be searched separately to produce the result for each of sub-query as a ranked list of highly ranked video.

2. QB also need to be integrated at each of sub-result by selecting the first frame of each top relevant shot for being verified in the SP step. The verified shot that pass the ADINT threshold will be regarded as a relevant.

3. The verified shots will be converted into a transaction regarding as all frames within a shot is of one transaction.



(A)



(B)

FIGURE 6.45: The evaluation result for (A) INS2011 and (B) INS2013 datasets were run with BoVW, AQE, QB, QBSP. As result, QBSP has lower mAP than the other methods.

4. Process QB as normally did on an image-type dataset to produce the final product as a second round query image.

5. Retrieving all the second round queries separately and fusing all the result by using max late fusion scoring scheme.

Therefore, we evaluate the performance of INS2011 dataset with standard BoVW, AQE, QB, and QBSP. And we evaluate the performance of INS2013 dataset with BoVW and QBSP. The evaluation result is reported in the figure 6.45. As result, our QBSP drop the overall performance, and much lower than all other methods on both INS2011 and INS2013. This imply that, we can find the limitation and the edge conditions where our QBSP cannot help improving the performance for these datasets.

To do that, we then look into more details at the topic level as shown in the figure 6.46a. According to the result, there is only one case (topic '9032') that QBSP win all the other methods, while mostly QBSP has lower APs than the others.

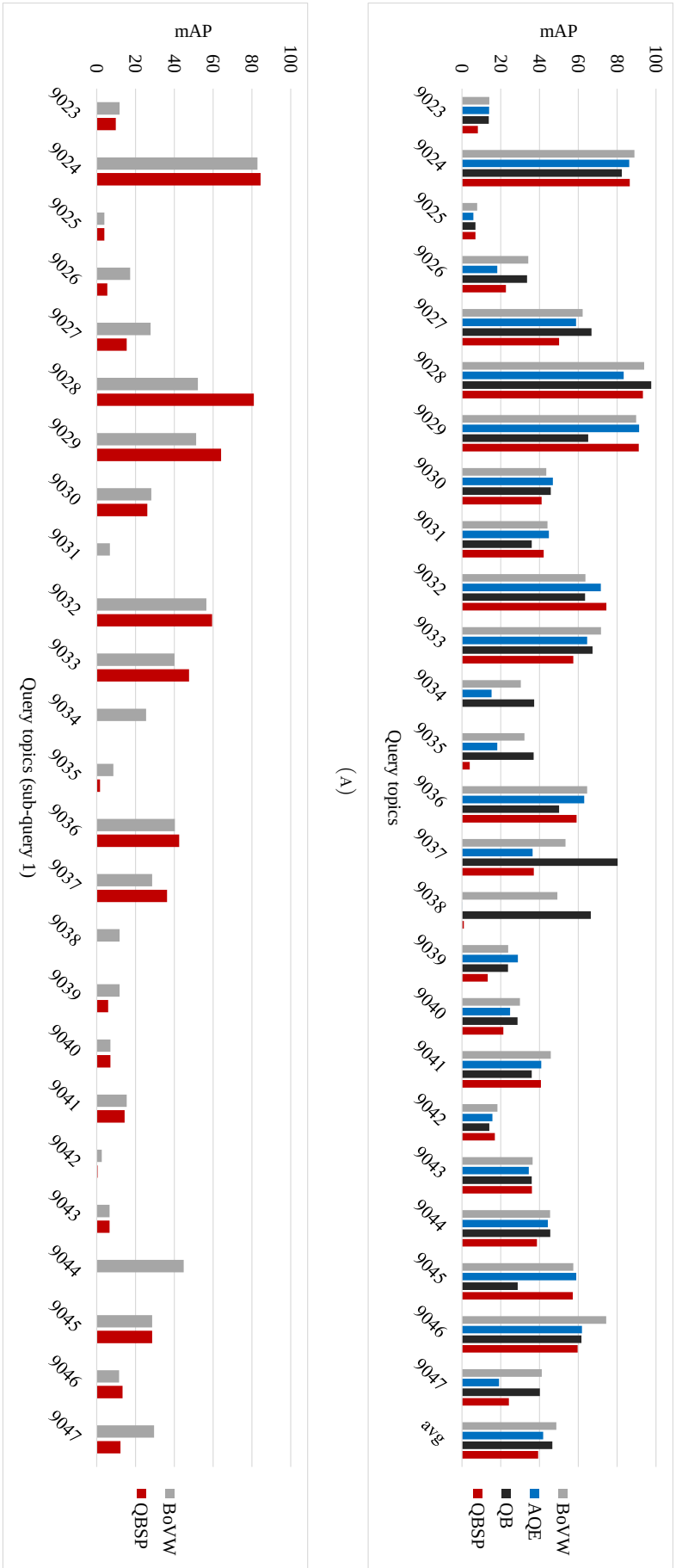


FIGURE 6.46: The performance evaluations on INS2011 dataset shows (A) The APs of BoVW, AQE, QB, and QBSP for each query topic. (B) The APs of BoVW and QBSP for a sub-query no.1 of each query topic.

However, we also look into more details, specifically at the sub-query no.1 for each topic, which we reported in the figure 6.46b. The result shown that, our QBSP wins on 8 topics (9024, 9028, 9029, 9032, 9033, 9036, 9037, and 9046) Even our method successfully improve the performance for these topics, however, these are just a small number comparing to all sub topic together. Thereby, it imply that these topics are important and will explain why our QBSP works and why it does not works. The examples of successful topic by using QBSP are shown in the figure 6.47 (9028), figure 6.48 (9029), and figure 6.49 (9033).

Working topic: 9028 The topic 9028 provides an airplane as a target query. As result (see figure 6.47), BoVW produced first round retrieval result with several big enough airplane. Also, the AP is quite good enough as 52.14%. Therefore, our QBSP will do mining with a correct focus on a target airplane, and finally improve the retrieval performance to 80.98%. Together as the background of this airplane are somehow support this airplane object. Thus, the learned context may also help in this query case.

Working topic: 9029 The topic 9029 provides a human as a target object. As result (see figure 6.48), BoVW produced first round retrieval result of with the same people that appear within the same room. This mean, QBSP will surely take the advantage of context discovery from the background as found to be the room texture, wall, etc. The result of BoVW show an AP is quite good enough at 51.26%. However, the target object is human that usually deformable and unluckily that this human wear the cloth that has less texture, the foreground itself will not help much in this retrieval, and also QBSP may regard this foreground as an object occlusion of the background. Therefore, QBSP improve this result a little to 64.12%, that probably come from the pure benefit of using the learned contexts.

Working topic: 9033 The topic 9033 provides a balloon as a target query. However, the target query is mostly occluded by the other balloons. As result (see figure 6.49), BoVW produced first round retrieval result with all the same environment as seen in the query image. The top retrieved images are consisted with 'that balloon', other balloons, the same grass field, the same view angle to the same mountain area. Also, at some result, the there are less area of occlusion which allow QBSP to learn more the hidden visual words of this target balloon. All these benefit together, Our methods boost the performance from BoVW as an AP of 40.07% to reach the final retrieval performance to 47.61%.

(A) Query

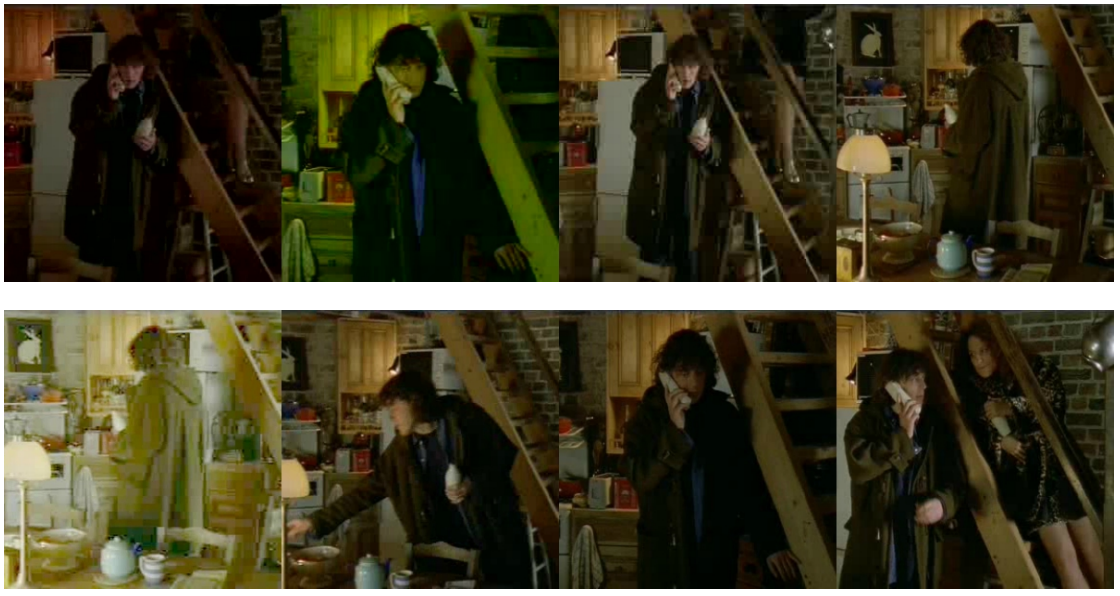
(C) BoVW: AP of 51.14%

(E) QBSP: AP of 80.98%

FIGURE 6.47: The working example of QB with INS2011 dataset (9028).

(A) Query

(C) BoVW: AP of 51.26%



(E) QBSP: AP of 64.12%

FIGURE 6.48: The working example of QB with INS2011 dataset (9029).

(A) Query

(C) BoVW: AP of 40.07%

(E) QBSP: AP of 47.61%

FIGURE 6.49: The working example of QB with INS2011 dataset (9033).

Most cases does not work However, in the most cases of the query topics show that QBSP does not work well with INS dataset. There are two main reason, one is there are no context to be able to learn on the top relevant images, second is the target object itself do not have much stable visual words enough to construct patterns. The example of this case is the topic 9035 as in the figure 6.50. The topic 9035 provides a turtle as the targeted query object. As result, BoVW returns much number of turtle images with high variations as an AP of 18.72%, so QBSP cannot find the enough patterns to infer the focus to the turtle and construct pattern correctly on it. Also, the background are rather different in all retrieved images. So, QBSP drops most of visual words and search for the second round with only AP of 3.85%.

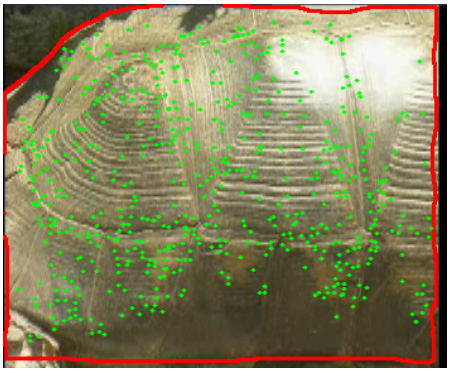
6.9.2.2 Target dataset characteristics

In this context, we then be able to summarize that, QB will work perfectly when original BoVW provides good enough result, e.g., some number of an object that visually similar to the query image, and it will boost much performance when the object is big enough and not too small. QB will also help improving the performance by using context, e.g., the query is to find an object that does not move such as a landmark, or to find an object that mostly appear together with some specific environment. So, the context information e.g., building besides, object beside, background, will help much in extending the description to the target object.

6.9.2.3 Weakness summarization

For the weakness of QB is that QB will not work if only one true positive result is provided, so there will be no object consistency to be discovered, which is the situation of the Stanford MVS dataset. Another thing is, QB cannot used to search for a deformable object, e.g., cloth, animal, texture less object etc. The mentioned situations are mostly belong as the characteristic of INS2011 and INS2013 dataset. So, that is the reason why QB works very well with Oxford and Paris dataset (landmark) and does not work with INS dataset.

Another thing to be mentioned is that, the retrieved results of QB are narrow in term of variation. This is because of QB try to find thing that similar to each others out of the given relevancies. Therefore, in case of the work that need to get the wider scope of the result, QB will not answer to the question.



(A) Query

(C) BoVW: AP of 18.72%

(E) QBSP: AP of 3.85%

FIGURE 6.50: The failure example of QB with INS2011 dataset (9035).

Chapter 7

Conclusions

"There is no real ending. It's just the place where you stop the story."

— Frank Herbert

In this chapter, we summarize the achievements of the work presented in this thesis and discuss in future research directions.

7.1 Achievements remark

In this thesis, we have proposed several methods for improving object-based image and video retrieval by major in relying on the use of data mining technique for discovering the consistency of patterns. The performance is evaluated considering on all standard datasets (Oxford 5k, Oxford 105k, and Paris 6k), including our newly extended datasets that merged MIR Flickr 1M together as a large distractor (Oxford 1M and Paris 1M). In chapter 3, we have proposed the way to integrate data mining technique, in help constructing the better representative query for the second round retrieval, which is known as relevance feedback or query expansion. Query Bootstrapping (QB) is proposed as a variant of Query Expansion (QE), to better exploit the correlation in the first-round retrievals. The aimed is to overcome the problem of the strong dependence on a query image in spatial verification in AQE. We employed a principled data mining tool, namely, frequent itemset mining (FIM), to find less ambiguous but meaningful co-occurring visual words (patterns) from the initial ranked list.

In the same chapter, we further presented an on-the-fly *support* tuning algorithm in section 3.3.2 for providing the best locally optimal *support* values for each of a query topic independently by regarding in tuning the best *minsup* and *maxsup* for the visual mining process. Although the idea is good and reasonable, these results have still not been topped, and may appeared to be below the former baseline, by the irrelevant of the retrieved images from BoVW itself.

In chapter 4, we proposed a proper way to utilize the spatial verification with our mining components. By integrating a LO-RANSAC algorithm for filtering the whole image rather than filtering deeply into the visual word level. Therefore, our QB can take much advantage as to mine the patterns with more focus to the target object. In addition, we presented an automated *inlier threshold* algorithm in section 4.3.2 for tuning an inlier threshold adaptively for LO-RANSAC spatial verification.

Since we proposed all the work together that aimed for finding the best pattern out of the relevant images. These patterns are regarded as the flag of frequent object candidate *fi*. Therefore, the patterns found by FIM are taken into account in a new weighting scheme called *tf-fi-idf*.

Moreover, in the chapter 5, we presented how to reduce the time in handling images with large number of visual words. Usually, these kind of transaction built up by using image words will caused too large number of patterns. By integrate the existing technique called transaction transposition will help us reduce the time in fining pattern quite significantly.

The overall steps of our proposed methods can be roughly go through as the following:

1. Baseline BoVW

The input query image is processed using the standard BoVW-based image retrieval to obtain the first-round result (Ranked List 1).

2. Spatial verification (SP, optional)

In this optional step, LO-RANSAC verifies the consistency of ranked list (returning a Verified Ranked List 1). Here, the spatial topology of the local features in each image in the Ranked List 1 is compared with that of the query image, and if the topologies are determined to be similar, the image is added to the verified list. The key parameter of LO-RANSAC, namely, the *inlier threshold*, is automatically determined.

3. Query Bootstrapping (QB)

The list (either verified or not verified) is fed to the FIM module in order to find frequently co-occurring visual words. The parameters of FIM, namely, *minsup*

and *maxsup*, are automatically determined. The result is used to determine *tf-fi-idf* weights, and the query vector for the second round is generated. Finally, second-round retrieval is executed with the generated second round query using a standard BoVW framework.

To the best of our knowledge, we are the first to investigate auto-tuning of the *support* parameter for FIM for a visual mining application and auto-tuning of the *inlier threshold* for LO-RANSAC geometric verifications. We presented head-to-head comparisons with state-of-the-art methods in terms of performance, robustness, and execution time. We evaluated our approach with the standard evaluation protocol on three well-known benchmark datasets, namely, Oxford 5k, Oxford 105k, and Paris 6k and also with our extended Oxford 1M and Paris 1M. The results show our approach has reached a new level of performance far beyond that of the classic BoVW framework or even AQE.

7.2 Future work

This section discusses some potentially promising directions for future work, as well as the speed issues in both a large scale object-based image and video retrieval and also the way to properly handling large number of patterns for which we believe it is important for researchers to solve.

One possibility to extend this research is that the way to proper handle each type of query properly. As we know there is a problem in the colossal patterns, if we can detect the task in advance e.g., "Hard" query type. The automatic or the hybrid model for AQE/QBSP will be much useful to reduce the overall time cunsumption taken by FIM module when colossal pattern is detected and then let AQE handles the task.

Another possibility is that integrating the binary feature is still good to be discovered with this work. We did some experiment and found that, the binary feature like ORB feature is contain some compliment ability to our current SIFT feature. For example, ORB is much faster in order to extract the feature from each image, while the actual descriptor is very lightweight comparing to the traditional SIFT. We did some test and we also found SIFT is good for landmark and the object that may have some perspective. In contrast, ORB feature is very good for the planar object like book cover cd/dvd cover. So, they are somehow compliment to each other, and it might be quite interesting to be discovered.

Appendix A

PVSS: Portable Visual Search Service for Researchers

A.1 Introduction

Image retrieval researchers aim to build good algorithms and devise methods to produce accurate ranked lists. However, the search results themselves appear rather repetitious, and the experience of viewing them is akin to listening to the same song 100 times a day. Imagine a better search in which the researcher has a ready-to-use demonstration interface for showing his results to audiences who can interactively send image queries and see the results effortlessly; this is our aim.

We present a Portable Visual Search Service for a researcher who desires to make his or her algorithm appear to be as "cool" as the best interactive demos at image retrieval conferences. Our service is a handy package, powered by VMWare virtualization technology, that is booted and served locally within a local network on a medium-spec laptop PC. The package contains all the necessary stuff to launch an on-site demo, e.g., a database, algorithm, and libraries. Moreover, our service is accessible using standard web browsers and client architectures, e.g., PC, iPhone, iPad, and Android.

This interactive system has been tested together with a large database at conferences and other events. The audience of a demo/poster session was impressed by the features of our service. In this context, our service handles the user interaction with the audience and gives researchers more freedom to explain their algorithm.

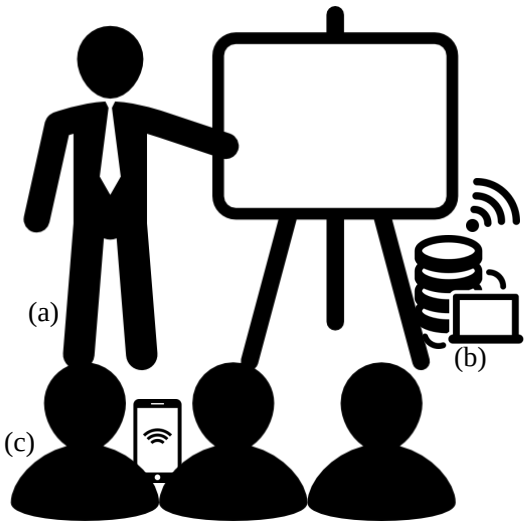


FIGURE A.1: A situation at the poster where a researcher (a) presents the work together with an on-site demo-ready (b) for serving image query from audiences (c).

A.1.1 Motivation

Research in the field of image retrieval seems delightful at first glance. New grad students inevitably expect that their research and results will tangibly reflect their vision. However, as experienced researchers know, studying image retrieval is not as easy as photo editing and is not interesting in terms of visual representation. Most of the time, the daily practice of research feels more like coding on 80s monochrome monitors and evaluations of retrieval results are akin to listening to the same music playlist over and over.

Firstly, researchers may have to examine several versions of branched algorithms. In order to see the retrieved images from each version that they have labored on over the holidays, they need to make use of a non-interactive tool, such as the image viewer in MATLAB or a dialog pop-up with OpenCV, or they may even manually open image files one-by-one to see what the result is. How does the prospect of checking for differences among several ranked lists containing a hundred images sound to you? Researchers regularly face problems in which the amount of returned images is far too large for manual checking. Manually selecting images is acceptable if the purpose is to insert a few figures in a publication. However, a smart researcher might write an extra piece of code to collage images together in one click. A bit more dynamic result can be had by generating a static HTML file for visualizing a ranked list on a web browser. Accordingly, a more sophisticated approach with a more dynamic flow will yield more impressive results as far as their appearance goes and would at least be a source of satisfaction for researchers trying to present their results.

Secondly, researchers often have to make oral presentations and demonstrate their algorithms. Here, what are the things they may need to show to audience? Static images? A flat poster? It should be the algorithm itself. And it should be in a way that impresses the audience. However, in conference sessions, they face crowds of strangers. A rather painful experience awaits those hoping to attract an audience with only static images on a poster. They may lose confidence and fail to convey the usefulness of their algorithm. The better solution would be running a working system that takes query images from audience members and returns the retrieved results directly to their mobiles so they can interactively explore what the researcher is trying to explain.

Thirdly, the result of building an interactive demo for a retrieval system will depend on the size of the database. For a small database, a laptop PC or a mobile device will be enough to run a standalone retrieval system. However, if the system has to be installed or configured on the user's device, this might be too bothersome for the audience and would take up the researcher's time by his or her providing support for different kinds of devices. Essentially, the only solution to this problem is to have less user configuration and the capability of handling a large database that cannot be fitted or compressed into a mobile device as in mobile visual search (MVS) [87, 88, 92, 93, 102]. Currently, this is only possible by hosting the system on a high-spec server and by using a standard web-based interface for handling image queries and visualizing the retrieved images. However, in going to an on-site demonstration, one faces several uncontrollable factors. One of those is of course whether or not an Internet connection will be available at the time of the presentation.

We propose the framework shown in figure A.2 as a way of ameliorating the above situations. The model of this framework handles two important aspects of demonstrating image retrieval research, i.e., taking query images and returning result images. The model is simple in that a researcher can easily connect their complex algorithm for core retrievals in a back-end fashion. Our framework has several sub-modules to handle a complete one-round retrieval, from taking an image query from a user and handing it to the API of the algorithm, to getting the retrieved result and visualizing it on the user's browser.

To enable the service even without an Internet connection, we exploited the power of virtualization technology, namely, VMWare, to pack the whole server configuration including an algorithm, a database, reference images, libraries, and even a web-server into one image file. Therefore, the researcher is able to serve a live demo with a minimum of requirements, i.e., all that is needed is his or her laptop with an Ethernet or wireless connection. The demo runs on a laptop equipped with either a Windows or Mac Os, and with Linux in the configured algorithm environment. Moreover, the framework is

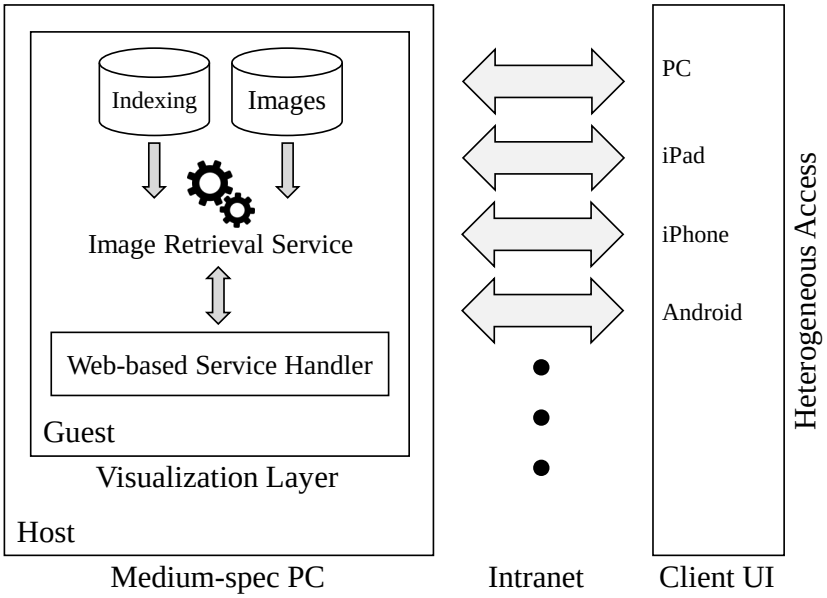


FIGURE A.2: Overall architecture of our framework consisting of an image retrieval service, which is connected to a web-based service handler that runs using a web server under a virtualization layer, a host computer, and a client web interface for heterogeneous devices.

designed for a client-server architecture with the use of web-server technology on a server and a standard web browser on a client. This combination makes it easy for audiences at conferences to try the presenter's algorithm on their own devices, as illustrated in Figure A.1.

Our paper is organized as follows. The related work is described in section A.1.2. The framework is described in section A.2, including the architecture of the server module and client module. The paper concludes with discussion and possible future work in section A.2.3.

A.1.2 Related work

Client-server architectures for image retrieval are implemented mostly on high-spec servers and with a specific application implemented in a platform-dependent programming language at the client.

There are a number of studies on client-server based image retrieval. Regarding those on large-scale databases, examples include the studies of Zhu et al. [78, 122], which used a database the size of half terabyte. On the other hand, some studies, e.g., [87, 88, 92, 93, 102], compress the database so it can be run directly on a mobile device; this leads to trade-offs in performance and information in the result.

We focus on the accuracy and detail of the meta-data, and this means we are interested in a large-scale database system with portability. Here, VMWare [153] and VirtualBox [154] are available as virtualization technologies . We chose VMWare in our current version on account of its performance.

A.2 PVSS architecture

We devised a portable visual search service (PVSS) that easily connects with a core image retrieval module. By utilizing the client-server architecture, a connection can be made using any available network, even without the Internet. The overall architecture of PVSS is shown in Figure A.2.

Our service framework has two major parts, the server side (section **??**) and the client side (section A.2.2).

A.2.1 Server modules

The server service is designed to be independent of the core retrieval module. However, the modules are easily connected with few configuration and data interchangeably rules.

Web-based Service handler The service handler is a connection point between a user request from the web-based client side and the core algorithm of the image retrieval module (Figure A.3).

Once a service is requested from a client browser, the web server establishes a connection with a specific session id followed by delivery of an HTML5 based web interface to the browser. The user is then able to submit a query using the provided options, which is handled at the server using the HTTP POST method. AJAX is used in order to wait for a response such as a retrieval result or error message.

On the server side, the connection between web-based service handle and image retrieval service is designed to be simple with no server modifications or reconfigurations. We implemented this connection based on a structural text file with delimiters. The connection is made by reading and writing files in a user space file system.

There are two files involved in transferring data, a query-specific file and a result-specific file. In the query-specific file, the image queries are queued for being read out and processed. The query file is named “queries.txt”, and is located on a specific pre-configured

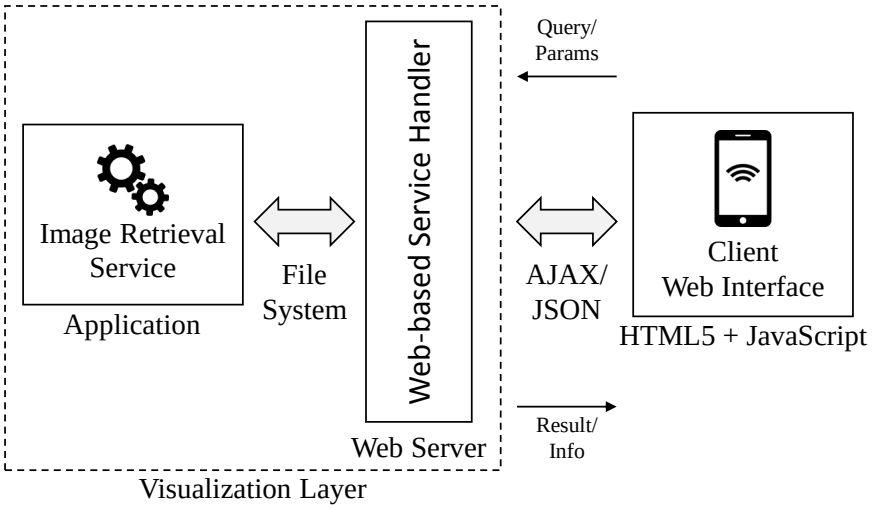


FIGURE A.3: Service connection model for transferring data between core retrieval module and client web interface.

path for both the web-based service handler and image retrieval module. The query template file has the following code 10:

```
query_image=path1|path2|...|pathN
result=directory
option=opt1|opt2|...|optM
$END$
query_image=path1|path2|...|pathN
result_path=directory
option=opt1|opt2|...|optM
$END$
...
```

LISTING A.1: Query template

The image retrieval module reads the above file according to the template.

- *query_image* is the image path, which can be allocated to multiple images at once with the delimiter '|'.
- *result* is a directory to the location where the retrieval result will be placed.
- *option* is an additional parameter that can be set by the client.

We encourage queries to be handled in parallel, if possible, so that they can be processed simultaneously.

For responding to a query, the result-specific file is provided for describing the retrieved images, and it has to be placed on a specified *result_path* with the file name "result.txt" for each query request. Once the web-based service handler detects an existing result

file, the corresponding data will be read out and sent back to the client. The template for "result.txt" is as follows:

```
status=message(text),
success(text),
error(text)
total_image=X
image=path
score=val
ext_info=info1|info2|...|infoY
image=path
score=val
ext_info=info1|info2|...|infoY
...
```

LISTING A.2: Result template

where *status* can be any kind of information for intermediately informing the user before the process finishes. For example, "status=message(Feature extraction..)" will be interpreted as an intermediate message to show with the client-side interface. Other examples include "status=error(Image too small)" and "status=success(Retrieval done!)".

total_image is the number of images in the result file. For each image, there is the field *image* specifying the image path, and a similarity score *score* for comparing it with the given query. Extra information can be embedded in the ext_info field.

Core image retrieval service The core image retrieval algorithm can be written in any programming language. In order to get the query request, researcher have to connect their module to the connection point provided by the web-based service handler. Additionally, the "queries.txt" file is read to process the individual query, and the corresponding "result.txt" is returned in a specific result directory.

Virtualization layer Portability issues arise when a live demo is unable to access the Internet. We put our framework on top of VMWare for hosting the system on a virtualization layer under any host machine. After the system is booted up, clients can access the service through the same network. An example of the best-case scenario (Figure A.4) is the presenter running a live demo on a computer connected to the conference network using a LAN cable, and the live demo is served to the audience' s devices seamlessly as the network also has WiFi.

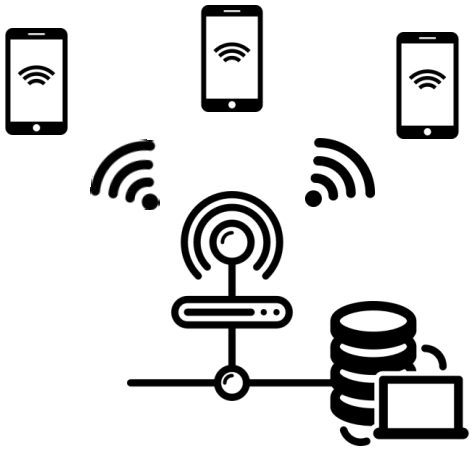


FIGURE A.4: Best case scenario where a live demo is hosted on a machine connected to a network using a wired connection and it is able to serve the audience via a wireless connection.

A.2.2 Client modules

Client side is designed to support heterogeneous client devices. We implemented the client service based on an HTML5 technology, which can be accessed by most recent PC and mobile browsers.

Client HTML5-based interface Figure A.5 and Figure A.6 respectively show examples of client interfaces for a PC and a mobile browser. The overall user interface is composed of two main sections, the query section (at the top of the web page) and result section (at the bottom of the web page).

The query section is to review the query that was submitted by the user. The core retrieval module can accept multiple query images for each retrieval, and the user can send all of them at once. The result section visualizes the retrieved images, which can be images, videos, and associated meta-data, such as a retrieval similarity score or a link to a video.

We also provide input options that may be useful to the operation of the core retrieval module. These can be parameters and specific input that need to be specified by the users, e.g., RANSAC on/off, debug mode on/off, etc.

Query and result handling component According to Figure A.3, the sent query is composed of images and parameters, and the received result is an image list with meta-data.

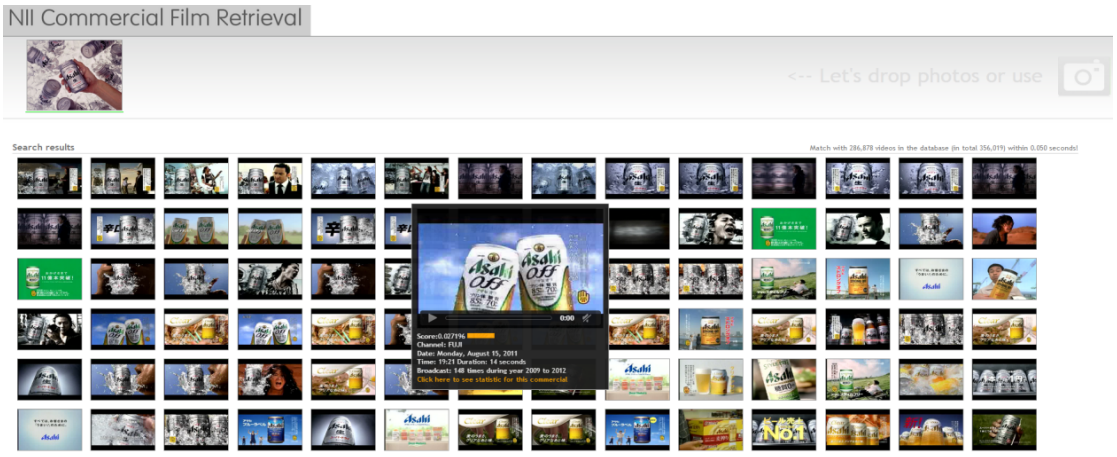


FIGURE A.5: Example of client-side user interface on a PC browser.

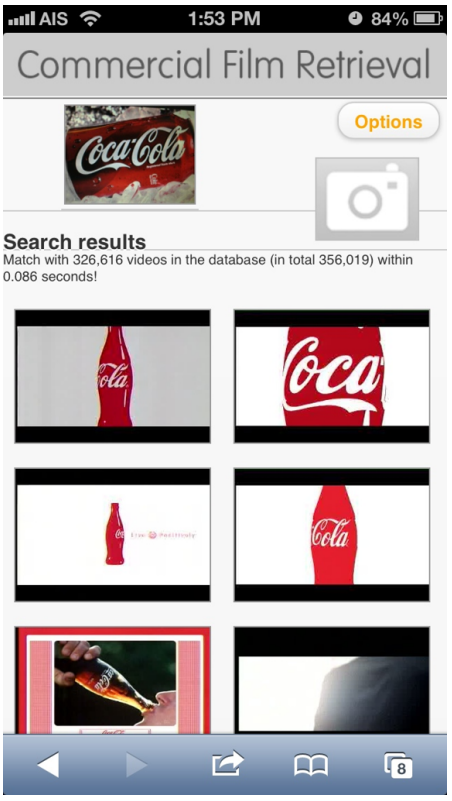


FIGURE A.6: Example of client-side user interface on a mobile browser.

For sending an image query, we provide several query sending modules based on the HTML5 web-RTC standard, including functionalities for drag-and-drop uploading of image(s) from the local machine (PC), taking a photo using the camera of a device (PC/mobile), and uploading images from a device (PC/mobile).

When the server is connected to a network with Internet access, the querying function is extended to external online resources, with functionalities such as dragging and dropping image URLs from other web sites (PC) and text-to-image retrieval (PC). We

implemented the text-to-image option by crawling images from Google image search using a specific keyword, then feeding the search result to the core retrieval module. This text-to-image functionality is an interesting option for using resources from the Internet instantly.

In order to transfer data from the client interface to the web-based service handler, we use AJAX to send a query as a binary stream and wait for the result. Contrastingly, the data sent from the web-based service handler will be in the form of an array of key-value pairs embedded in a JSON string. The format of this JSON result is based on the format of the result template.

Lastly, in order to interpret the retrieved results in the result area, the client interface waits for a "status=success()" as an acknowledgment to start interpreting the results and meta-data. Otherwise, the status message or an error message will be shown as an intermediate response telling the user what is going on.

A.2.3 Conclusion

We proposed an interactive service framework of image retrieval for a researcher to easily visualize his or her retrieval results, test systems with real-world query data, and to present algorithms in operation at the conference. The system mainly provides a service for off-Internet live demos; it has high portability and can be served anywhere. The system can also easily access the Internet resources where such an environment exists.

We implemented s client interface based on the HTML5 standard that is supported by recent devices like PCs, iPhones, iPads, and Android devices and is familiar to any audience.

Our system is currently good for audiences sending queries and receiving results. However, if the traffic generated by the audience is high, or higher frame rates or low latency is needed, our system should be integrated with WebSockets to create a connection between each client and the server; this is our future work.

We encourage readers to follow-up on our website[1] for the official release of our PVSS project.

[1] http://www.satoh-lab.nii.ac.jp/~stylix

Bibliography

[1] Instagram. Instagram developper documentation. URL https://www.instagram.com/developer/.

[2] Yahoo. Flickr services. URL https://www.flickr.com/services/api/.

[3] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[4] F. W. Lancaster and E. G. Fayen. *Information retrieval on-line*. 1973.

[5] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975. ISSN 0001-0782. doi: 10.1145/361219.361220. URL http://doi.acm.org/10.1145/361219.361220.

[6] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513 – 523, 1988. ISSN 0306-4573. doi: http://dx.doi.org/10.1016/0306-4573(88)90021-0. URL http://www.sciencedirect.com/science/article/pii/0306457388900210.

[7] Stanford Vision Lab, Stanford University, and Princeton University. Imagenet. URL http://www.image-net.org/.

[8] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38 (11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL http://doi.acm.org/10.1145/219717.219748.

[9] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003. URL http://doi.ieeecomputersociety.org/10.1109/ICCV.2003.1238663.

[10] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531 vol. 2, June 2005. doi: 10.1109/CVPR.2005.16. URL https://dx.doi.org/10.1109%2FCVPR.2005.16.

[11] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410. URL https://dx.doi.org/10.1109%2FICCV.1999.790410.

[12] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, pages 128–142, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43745-2. URL http://dl.acm.org/citation.cfm?id=645315.649184.

[13] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, pages 63–86, 2004. ISSN 0920-5691. URL http://dx.doi.org/10.1023/B%3AVISI.0000027790.02288.f2.

[14] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, November 2005. ISSN 0920-5691. doi: 10.1007/s11263-005-3848-x. URL http://dx.doi.org/10.1007/s11263-005-3848-x.

[15] Jiří Matas Michal Perďoch, Ondřej Chum. Efficient representation of local geometry for large scale object retrieval. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9–16, June 2009. ISBN 978-1-4244-3994-2. URL http://dx.doi.org/10.1109/CVPRW.2009.5206529.

[16] Jiří Matas Michal Perďoch, Ondřej Chum. Hessian-affine detector with sift descriptor. URL http://cmp.felk.cvut.cz/~perdom1/hesaff/.

[17] J. MacQueen. Some methods for classification and analysis of multivariate observations. 1967. URL https://zbmath.org/?format=complete&q=an:0214.46201.

[18] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition, 1975. ISBN 047135645X. URL http://dl.acm.org/citation.cfm?id=540298.

[19] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001. ISSN 0920-5691. doi: 10.1023/A:1011126920638. URL http://dx.doi.org/10.1023/A%3A1011126920638.

[20] Jianxin Wu, Wei-Chian Tan, and James M. Rehg. Efficient and effective visual codebook generation using additive kernels. *J. Mach. Learn. Res.*, 12:3097–3118, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078205.

[21] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In Sandip Das and Ryuhei Uehara, editors, *WALCOM: Algorithms and Computation*, volume 5431 of *Lecture Notes in Computer Science*, pages 274–285. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00201-4. doi: 10.1007/978-3-642-00202-1_24. URL http://dx.doi.org/10.1007/978-3-642-00202-1_24.

[22] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, pages 1–8, 2007. URL http://dx.doi.org/10.1109/CVPR.2007.383172.

[23] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. A library for fast, distributed clustering. URL https://github.com/philbinj/fastcluster.

[24] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, pages 331–340, 2009. URL http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-222.

[25] Marius Muja. Fast library for approximate nearest neighbors. URL https://github.com/mariusmuja/flann.

[26] Andrej Mikulík, Michal Perdoch, Ondrej Chum, and Jiri Matas. Learning a fine vocabulary. In *ECCV*, pages 1–14, 2010. URL http://dx.doi.org/10.1007/978-3-642-15558-1_1.

[27] Over Paul, Fiscus Jon, Sanders Greg, Joy David, and Michel Martial. Trecvid 2014 – an overview of the goals, tasks, data, evaluation mechanisms, and metrics. NIST, USA, 2014. URL http://www-nlpir.nist.gov/projects/tvpubs/tv14.papers/tv14overview.pdf.

[28] Duy-Dinh Le, Vinh-Tiep Nguyen, Cai-Zhi Zhu, Duc M. Nguyen, Thanh Duc Ngo, Siriwat Kasamwattanarote, Poullot Sebastien, Minh-Triet Tran, Duc A. Duong, and Shin'ichi Satoh. Nii at trecvid 2014 instance search task. In *TREC Video Retrieval Evaluation: TRECVID*, 2014. URL http://www-nlpir.nist.gov/projects/tvpubs/tv14.papers/nii.pdf.

[29] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004. ISSN 0920-5691. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[30] Relja Arandjelovic. Three things everyone should know to improve object retrieval. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 2911–2918, Washington, DC, USA, 2012.

IEEE Computer Society. ISBN 978-1-4673-1226-4. URL http://dl.acm.org/citation.cfm?id=2354409.2355123.

[31] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, pages 355–364, 1990. URL http://dl.acm.org/citation.cfm?id=275537.275712.

[32] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, pages 1–8, 2007. URL http://dx.doi.org/10.1109/ICCV.2007.4408891.

[33] Karel Lebeda, Ji Matas, and Ondrej Chum. Fixing the locally optimized RANSAC. In *BMVC*, pages 1–11, 2012. URL http://dx.doi.org/10.5244/C.26.95.

[34] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993. ISSN 0163-5808. doi: 10.1145/170036.170072. URL http://doi.acm.org/10.1145/170036.170072.

[35] Feida Zhu, Xifeng Yan, Jiawei Han, Philip S. Yu, and Hong Cheng. Mining colossal frequent patterns by core pattern fusion. In *ICDE*, pages 706–715, 2007. URL http://dx.doi.org/10.1109/ICDE.2007.367916.

[36] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. Lcm ver.1: An efficient algorithm for enumerating closed patterns in transaction databases. In Einoshin Suzuki and Setsuo Arikawa, editors, *Discovery Science, FIMI*, volume 3245 of *Lecture Notes in Computer Science*, pages 16–31. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-23357-2. doi: 10.1007/978-3-540-30214-8_2. URL http://dx.doi.org/10.1007/978-3-540-30214-8_2.

[37] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, pages 1–8, 2008. URL http://dx.doi.org/10.1109/CVPR.2008.4587635.

[38] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*, MIR '08, pages 39–43, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-312-9. doi: 10.1145/1460096.1460104. URL http://doi.acm.org/10.1145/1460096.1460104.

[39] Ondrej Chum, Andrej Mikulik, Michal Perdoch, and Jiri Matas. Total recall II: Query expansion revisited. In *CVPR*, pages 889–896, 2011. URL http://dx.doi.org/10.1109/CVPR.2011.5995601.

[40] Danfeng Qin, S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 777–784, June 2011. doi: 10.1109/CVPR.2011.5995373. URL http://dx.doi.org/10.1109/CVPR.2011.5995373.

[41] Yanzhi Chen, Xi Li, A. Dick, and A. van den Hengel. Boosting object retrieval with group queries. *Signal Processing Letters, IEEE*, 19(11):765–768, Nov 2012. ISSN 1070-9908. doi: 10.1109/LSP.2012.2216875. URL http://dx.doi.org/10.1109/LSP.2012.2216875.

[42] Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, pages 1958–1970, 2008. URL http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.128.

[43] Xiao Zhang, Zhiwei Li, Lei Zhang, Wei-Ying Ma, and Heung-Yeung Shum. Efficient indexing for large scale visual search. In *ICCV*, pages 1103–1110, 2009. URL http://dx.doi.org/10.1109/ICCV.2009.5459354.

[44] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 36.1–36.10. BMVA Press, 2002. ISBN 1-901725-19-7. URL http://dx.doi.org/10.5244/C.16.36. doi:10.5244/C.16.36.

[45] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1508–1515, Washington, DC, USA, October 2005. IEEE Computer Society. ISBN 0-7695-2334-X-02. doi: 10.1109/ICCV.2005.104. URL http://dx.doi.org/10.1109/ICCV.2005.104.

[46] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010. doi: 10.1109/TPAMI.2008.275. URL http://lanl.arXiv.org/pdf/0810.2434.

[47] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[48] Zhiyong Ye, Yijian Pei, and Jihong Shi. An improved algorithm for harris corner detection. In *Image and Signal Processing, 2009. CISP '09. 2nd International Congress on*, pages 1–4, Oct 2009. doi: 10.1109/CISP.2009.5304635. URL http://dx.doi.org/10.1109/CISP.2009.5304635.

[49] Wikipedia. Scale-invariant feature transform. URL https://en.wikipedia.org/wiki/Scale-invariant_feature_transform.

[50] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005. URL http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.8745.

[51] Siriwat Kasamwattanarote, Nagul Cooharojananone, Shin' ichi Satoh, and Rajalida Lipikorn. Real time tunnel based video summarization using direct shift collision detection. In Guoping Qiu, KinMan Lam, Hitoshi Kiya, Xiang-Yang Xue, C.-C.Jay Kuo, and MichaelS. Lew, editors, *Advances in Multimedia Information Processing - PCM 2010*, volume 6297 of *Lecture Notes in Computer Science*, pages 136–147. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15701-1. doi: 10.1007/978-3-642-15702-8_13. URL http://dx.doi.org/10.1007/978-3-642-15702-8_13.

[52] N. Cooharojananone, Siriwat Kasamwattanarote, S. Satoh, and R. Lipikorn. Real time trajectory search in video summarization using direct distance transform. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 932–935, Oct 2010. doi: 10.1109/ICOSP.2010.5655723. URL http://dx.doi.org/10.1109/ICOSP.2010.5655723.

[53] Nagul Cooharojananone, Siriwat Kasamwattanarote, Rajalida Lipikorn, and Shin' ichi Satoh. Automated real-time video surveillance summarization framework. *Journal of Real-Time Image Processing*, 10(3):513–532, 2015. ISSN 1861-8200. doi: 10.1007/s11554-012-0280-7. URL http://dx.doi.org/10.1007/s11554-012-0280-7.

[54] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014. URL http://dx.doi.org/10.1016/j.cviu.2007.09.014.

[55] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15560-X, 978-3-642-15560-4. URL http://dl.acm.org/citation.cfm?id=1888089.1888148.

[56] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126542. URL http://dx.doi.org/10.1109/ICCV.2011.6126542.

[57] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126544. URL http://dx.doi.org/10.1109/ICCV.2011.6126544.

[58] OpenCV. Opencv: Open sourse computer vision. URL http://opencv.org.

[59] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci.*, 2(2):438–469, April 2009. ISSN 1936-4954. doi: 10.1137/080732730. URL http://dx.doi.org/10.1137/080732730.

[60] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1469–1472, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-933-6. doi: 10.1145/1873951.1874249. URL http://doi.acm.org/10.1145/1873951.1874249.

[61] VLFeat. Vlfeat: an open source computer vision algorithms. URL http://www.vlfeat.org/.

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[63] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[64] Danfeng Qin, C. Wengert, and L. Van Gool. Query adaptive similarity for large scale object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2013*

IEEE Conference on, pages 1610–1617, June 2013. doi: 10.1109/CVPR.2013.211. URL http://dx.doi.org/10.1109/CVPR.2013.211.

[65] Panu Turcot and David G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV*, pages 2109–2116, 2009. ISBN 978-1-4244-4442-7. URL http://dx.doi.org/10.1109/ICCVW.2009.5457541.

[66] Pedro Henrique Bugatti, Marcela Xavier Ribeiro, Agma Juci Machado Traina, and Caetano Traina Jr. Content-based retrieval of medical images by continuous feature selection. In *Computer-Based Medical Systems, CBMS*, pages 272–277, 2008. URL http://dx.doi.org/10.1109/CBMS.2008.82.

[67] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages 1458–1465, 2005. URL http://doi.ieeecomputersociety.org/10.1109/ICCV.2005.239.

[68] Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, pages 71–84, 2010. ISBN 3-642-15554-5, 978-3-642-15554-3. URL http://dl.acm.org/citation.cfm?id=1888150.1888157.

[69] Christian Wallraven, Barbara Caputo, and Arnulf B. A. Graf. Recognition with local features: the kernel recipe. In *ICCV*, pages 257–264, 2003. URL http://doi.ieeecomputersociety.org/10.1109/ICCV.2003.1238351.

[70] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, 2006. ISBN 0-7695-2597-0. URL http://dx.doi.org/10.1109/CVPR.2006.264.

[71] Vijay R. Chandrasekhar, David M. Chen, Sam S. Tsai, Ngai-Man Cheung, Huizhong Chen, Gabriel Takacs, Yuriy Reznik, Ramakrishna Vedantham, Radek Grzeszczuk, Jeff Bach, and Bernd Girod. The stanford mobile visual search data set. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, pages 117–122, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0518-1. doi: 10.1145/1943552.1943568. URL http://doi.acm.org/10.1145/1943552.1943568.

[72] NIST. Trec video retrieval evaluation: Trecvid. URL http://trecvid.nist.gov/.

[73] Cai-Zhi Zhu and Shin'ichi Satoh. Large vocabulary quantization for searching instances from videos. In *ICMR*, pages 1–8, 2012. ISBN 978-1-4503-1329-2. URL http://doi.acm.org/10.1145/2324796.2324856.

[74] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *Int. J. Comput. Vision*, 87(3):316–336, May 2010. ISSN 0920-5691. doi: 10.1007/s11263-009-0285-2. URL http://dx.doi.org/10.1007/s11263-009-0285-2.

[75] Ameesh Makadia. Feature tracking for wide-baseline image retrieval. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 310–323. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15554-3. doi: 10.1007/978-3-642-15555-0_23. URL http://dx.doi.org/10.1007/978-3-642-15555-0_23.

[76] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *CVPR*, pages 1169–1176, 2009. URL http://dx.doi.org/10.1109/CVPRW.2009.5206609.

[77] Cai-Zhi Zhu, Siriwat Kasamwattanarote, Xiaomeng Wu, and Shin'ichi Satoh. Connect commercial films with realities. In *ICMR*, pages 323–324, 2013. ISBN 978-1-4503-2033-7. URL http://doi.acm.org/10.1145/2461466.2461527.

[78] Cai-Zhi Zhu, Siriwat Kasamwattanarote, Xiaomeng Wu, and Shin'ichi Satoh. Tell me about tv commercials of this product. In *MMM*, pages 242–253, 2014. URL http://dx.doi.org/10.1007/978-3-319-04114-8_21.

[79] X. Wu and S. Satoh. Ultrahigh-speed tv commercial detection, extraction, and matching. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23 (6):1054–1069, June 2013. ISSN 1051-8215. doi: 10.1109/TCSVT.2013.2248991. URL http://dx.doi.org/10.1109/TCSVT.2013.2248991.

[80] Henrik Stewénius, SteinarH. Gunderson, and Julien Pilet. Size matters: Exhaustive geometric verification for image retrieval accepted for eccv 2012. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, volume 7573 of *Lecture Notes in Computer Science*, pages 674–687. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33708-6. doi: 10.1007/978-3-642-33709-3_48. URL http://dx.doi.org/10.1007/978-3-642-33709-3_48.

[81] Apache Hadoop. Apache hadoop, an open-source software framework distributed storage and distributed processing. URL https://hadoop.apache.org/.

[82] Siriwat Kasamwattanarote, Yusuke Uchida, and Shin'ichi Satoh. Query bootstrapping: A visual mining based query expansion. In *IEICE Transactions on*

Information and Systems, volume Vol.E99-D, pages 454–466, February 2016. URL http://www.satoh-lab.nii.ac.jp/~stylix/.

[83] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2007. ISBN 9780136006633. URL http://dl.acm.org/citation.cfm?id=1410217.

[84] Douglas W. Jones. Page replacement, 22c:116, lecture notes. URL http://homepage.cs.uiowa.edu/~jones/opsys/fall95/notes/0908.html.

[85] Paul Torrez, Tim Hallner, Kiran Mathrani, and Anu Bhaskar. Virtual memory, lecture notes. URL http://www.read.cs.ucla.edu/111/2006fall/notes/lec11.

[86] Siriwat Kasamwattanarote and Shin'ichi Satoh. Pvss: Portable visual search service for researchers. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, ICIMCS '15, pages 64:1–64:5, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3528-7. doi: 10.1145/2808492.2808556. URL http://doi.acm.org/10.1145/2808492.2808556.

[87] Bernd Girod, Vijay Chandrasekhar, David M. Chen, Ngai-Man Cheung, Radek Grzeszczuk, Yuriy A. Reznik, Gabriel Takacs, Sam S. Tsai, and Ramakrishna Vedantham. Mobile visual search. In *ACMMM*, pages 117–122, 2011. ISBN 978-1-4503-0518-1. URL http://doi.acm.org/10.1145/1943552.1943568.

[88] Vijay Chandrasekhar, David M. Chen, Andy Lin, Gabriel Takacs, Sam S. Tsai, Ngai-Man Cheung, Yuriy A. Reznik, Radek Grzeszczuk, and Bernd Girod. Comparison of local feature descriptors for mobile visual search. In *ICIP*, pages 3885–3888, 2010. URL http://dx.doi.org/10.1109/ICIP.2010.5649937.

[89] D.M. Chen and B. Girod. Memory-efficient image databases for mobile visual search. *MultiMedia, IEEE*, 21(1):14–23, Jan 2014. ISSN 1070-986X. doi: 10.1109/MMUL.2013.46. URL http://dx.doi.org/10.1109/MMUL.2013.46.

[90] Vijay Chandrasekhar, Gabriel Takacs, David M. Chen, Sam S. Tsai, Yuriy Reznik, Radek Grzeszczuk, and Bernd Girod. Compressed histogram of gradients: A low-bitrate descriptor. *International Journal of Computer Vision*, pages 384–399, 2012. ISSN 0920-5691. URL http://dx.doi.org/10.1007/s11263-011-0453-z.

[91] Rongrong Ji, Ling-Yu Duan, Jie Chen, Hongxun Yao, Yong Rui, Shih-Fu Chang, and Wen Gao. Towards low bit rate mobile visual search with multiple-channel coding. In *ACMMM*, pages 573–582, 2011. ISBN 978-1-4503-0616-4. URL http://doi.acm.org/10.1145/2072298.2072372.

[92] Yu-Chuan Su, Tzu-Hsuan Chiu, Yan-Ying Chen, Chun-Yen Yeh, and Winston H. Hsu. Enabling low bitrate mobile visual recognition: A performance versus bandwidth evaluation. In *ACMMM*, pages 73–82, 2013. ISBN 978-1-4503-2404-5. URL http://doi.acm.org/10.1145/2502081.2502110.

[93] Jayaguru Panda, Michael S. Brown, and C. V. Jawahar. Offline mobile instance retrieval with a small memory footprint. In *ICCV*, pages 1257–1264, 2013. URL http://dx.doi.org/10.1109/ICCV.2013.159.

[94] Lican Dai, Huanjing Yue, Xiaoyan Sun, and Feng Wu. Imshare: Instantly sharing your mobile landmark images by search-based reconstruction. In *ACMMM*, pages 579–588, 2012. ISBN 978-1-4503-1089-5. URL http://doi.acm.org/10.1145/2393347.2393428.

[95] Junfeng He, Jinyuan Feng, Xianglong Liu, Tao Cheng, Tai-Hsu Lin, Hyunjin Chung, and Shih-Fu Chang. Mobile product search with bag of hash bits and boundary reranking. In *CVPR*, pages 3005–3012, 2012. URL http://dx.doi.org/10.1109/CVPR.2012.6248030.

[96] CloudSight. Camfind app - visual search & image recognition api powered by cloudsight. URL http://camfindapp.com.

[97] KDDI R&D Laboratories. Satch viewer - スマートフォン用新感覚arコミュニケーションツール. URL http://viewer.satch.jp.

[98] Florent Perronnin, Yan Liu, Jorge Sánchez, and Herve Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, pages 3384–3391, 2010. URL http://dx.doi.org/10.1109/CVPR.2010.5540009.

[99] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311, June 2010. doi: 10.1109/CVPR.2010.5540039. URL http://dx.doi.org/10.1109/CVPR.2010.5540039.

[100] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *PAMI*, pages 1704–1716, 2012. ISSN 0162-8828. URL http://dx.doi.org/10.1109/TPAMI.2011.235.

[101] Jorge Sánchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, pages 1665–1672, 2011. URL http://dx.doi.org/10.1109/CVPR.2011.5995504.

[102] David Chen, Sam Tsai, Vijay Chandrasekhar, Gabriel Takacs, Ramakrishna Vedantham, Radek Grzeszczuk, and Bernd Girod. Residual enhanced visual vector as a compact signature for mobile visual search. *Signal Process.*, 93(8):2316–2327, August 2013. ISSN 0165-1684. doi: 10.1016/j.sigpro.2012.06.005. URL http://dx.doi.org/10.1016/j.sigpro.2012.06.005.

[103] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, pages 304–317, 2008. ISBN 978-3-540-88681-5. URL http://dx.doi.org/10.1007/978-3-540-88682-2_24.

[104] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33 (1):117–128, Jan 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.57. URL http://dx.doi.org/10.1109/TPAMI.2010.57.

[105] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Packing bag-of-features. In *ICCV*, pages 2357–2364, 2009. URL http://dx.doi.org/10.1109/ICCV.2009.5459419.

[106] Stefan Romberg, Moritz August, Christian X. Ries, and Rainer Lienhart. Robust feature bundling. In *PCM*, pages 45–56, 2012. URL http://dx.doi.org/10.1007/978-3-642-34778-8_5.

[107] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, pages 1–10, 2008. URL http://dx.doi.org/10.5244/C.22.50.

[108] Ondrej Chum, Michal Perdoch, and Jiri Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, pages 17–24, 2009. URL http://dx.doi.org/10.1109/CVPRW.2009.5206531.

[109] Ondrej Chum and Jiri Matas. Unsupervised discovery of co-occurrence in sparse high dimensional data. In *CVPR*, pages 3416–3423, 2010. URL http://dx.doi.org/10.1109/CVPR.2010.5539997.

[110] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-8132-2. URL http://dl.acm.org/citation.cfm?id=829502.830043.

[111] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Sequential projection learning for hashing with compact codes. In *ICML*, pages 1127–1134, 2010. URL http://www.icml2010.org/papers/178.pdf.

[112] Zhong Wu, Qifa Ke, M. Isard, and Jian Sun. Bundling features for large scale partial-duplicate web image search. In *CVPR*, pages 25–32, 2009. URL http://dx.doi.org/10.1109/CVPRW.2009.5206566.

[113] Xiaoyu Wang, Ming Yang, T. Cour, Shenghuo Zhu, Kai Yu, and T.X. Han. Contextual weighting for vocabulary tree based image retrieval. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 209–216, Nov 2011. doi: 10.1109/ICCV.2011.6126244. URL http://dx.doi.org/10.1109/ICCV.2011.6126244.

[114] Shiliang Zhang, Qingming Huang, Gang Hua, Shuqiang Jiang, Wen Gao, and Qi Tian. Building contextual visual vocabulary for large-scale image applications. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 501–510, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-933-6. doi: 10.1145/1873951.1874018. URL http://doi.acm.org/10.1145/1873951.1874018.

[115] David C. Lee, Qifa Ke, and Michael Isard. Partition min-hash for partial duplicate image discovery. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 648–662, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15548-0, 978-3-642-15548-2. URL http://dl.acm.org/citation.cfm?id=1886063.1886113.

[116] Bangpeng Yao and Fei-Fei Li. Grouplet: A structured image representation for recognizing human and object interactions. In *CVPR*, pages 9–16, 2010. URL http://dx.doi.org/10.1109/CVPR.2010.5540234.

[117] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010. URL http://dx.doi.org/10.1109/CVPR.2010.5540018.

[118] Yang Cao, Changhu Wang, Zhiwei Li, Liqing Zhang, and Lei Zhang. Spatial-bag-of-features. In *CVPR*, pages 3352–3359, 2010. URL http://dx.doi.org/10.1109/CVPR.2010.5540021.

[119] Wengang Zhou, Yijuan Lu, Houqiang Li, Yibing Song, and Qi Tian. Spatial coding for large scale partial-duplicate web image search. In *ACMMM*, pages 511–520, 2010. ISBN 978-1-60558-933-6. URL http://doi.acm.org/10.1145/1873951.1874019.

[120] Yimeng Zhang, Zhaoyin Jia, and Tsuhan Chen. Image retrieval with geometry-preserving visual phrases. In *CVPR*, pages 809–816, 2011. URL http://dx.doi.org/10.1109/CVPR.2011.5995528.

[121] Junsong Yuan, Ying Wu, and Ming Yang. Discovery of collocation patterns: from visual words to visual phrases. In *CVPR*, pages 1–8, 2007. URL http://dx.doi.org/10.1109/CVPR.2007.383222.

[122] Cai-Zhi Zhu, Hervé Jégou, and Shin'Ichi Satoh. Query-adaptive asymmetrical dissimilarities for visual object retrieval. In *ICCV - International Conference on Computer Vision*, pages 1705–1712, 2013. URL http://dx.doi.org/10.1109/ICCV.2013.214.

[123] Nadav Ben-Haim, Boris Babenko, and Serge Belongie. Improving web-based image search via content based clustering. pages 106–106, Washington, DC, USA, 2006. ISBN 0-7695-2646-2. URL http://dx.doi.org/10.1109/CVPRW.2006.100.

[124] Josip Krapac, Moray Allan, Jakob J. Verbeek, and Frederic Jurie. Improving web image search results using query-relative classifiers. In *CVPR*, pages 1094–1101, 2010. URL http://dx.doi.org/10.1109/CVPR.2010.5540092.

[125] Relja Arandjelovic and Andrew Zisserman. Multiple queries for large scale specific object retrieval. In *BMVC*, pages 1–11, 2012. URL http://dx.doi.org/10.5244/C.26.92.

[126] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994. ISBN 1-55860-153-8. URL http://dl.acm.org/citation.cfm?id=645920.672836.

[127] Chan Man Kuok, Ada Fu, and Man Hon Wong. Mining fuzzy association rules in databases. *SIGMOD Rec.*, 27(1):41–46, March 1998. ISSN 0163-5808. doi: 10.1145/273244.273257. URL http://doi.acm.org/10.1145/273244.273257.

[128] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000. ISSN 0163-5808. doi: 10.1145/335191.335372. URL http://doi.acm.org/10.1145/335191.335372.

[129] Christian Borgelt. An implementation of the fp-growth algorithm. In *OSDM*, pages 1–5, 2005. ISBN 1-59593-210-0. URL http://doi.acm.org/10.1145/1133905.1133907.

[130] Christian Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012. ISSN 1942-4795. doi: 10.1002/widm.1074. URL http://dx.doi.org/10.1002/widm.1074.

[131] Takeaki Uno, Yuzo Uchida, and Hiroki Arimura. Lcm ver.2: An efficient algorithm for enumerating closed patterns in transaction databases. In *Discovery Science,*

FIMI, Lecture Notes in Computer Science. 2004. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.495.4958&rep=rep1&type=pdf.

[132] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, OSDM '05, pages 77–86, New York, NY, USA, 2005. ACM. ISBN 1-59593-210-0. doi: 10.1145/1133905.1133916. URL http://doi.acm.org/10.1145/1133905.1133916.

[133] Philip S. Yu, Xifeng Yan, Jiawei Han, Hong Cheng, and Feida Zhu. Approximate frequent pattern mining. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.7415.

[134] Jilles Vreeken, Matthijs Leeuwen, and Arno Siebes. Krimp: Mining itemsets that compress. *Data Min. Knowl. Discov.*, pages 169–214, 2011. URL http://dx.doi.org/10.1007/s10618-010-0202-x.

[135] Peter D. Grunwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[136] Yeong-Chyi Lee, Tzung-Pei Hong, and Wen-Yang Lin. Mining association rules with multiple minimum supports using maximum constraints. In *IJAR*, pages 44–54, 2005. doi: http://dx.doi.org/10.1016/j.ijar.2004.11.006. URL http://www.sciencedirect.com/science/article/pii/S0888613X04001392.

[137] Christian Borgelt. Frequent pattern mining lecture. URL http://www.borgelt.net/teach/fpm.

[138] T. Quack, V. Ferrari, B. Leibe, and L. Van Gool. Efficient mining of frequent and distinctive feature configurations. In *ICCV*, pages 1–8, 2007. URL http://dx.doi.org/10.1109/ICCV.2007.4408906.

[139] Sebastian Nowozin, Koji Tsuda, Takeaki Uno, Taku Kudo, and Gokhan H. Bakir. Weighted substructure mining for image analysis. In *CVPR*, pages 1–8, 2007. URL http://dx.doi.org/10.1109/CVPR.2007.383171.

[140] Andrew Gilbert, John Illingworth, and Richard Bowden. Fast realistic multi-action recognition using mined dense spatio-temporal features. In *ICCV*, pages 925–931, 2009. URL http://dx.doi.org/10.1109/ICCV.2009.5459335.

[141] Basura Fernando, Elisa Fromont, and Tinne Tuytelaars. Effective use of frequent itemset mining for image classification. In *ECCV*, pages 214–227, 2012. ISBN 978-3-642-33717-8. URL http://dx.doi.org/10.1007/978-3-642-33718-5_16.

[142] Till Quack, Vittorio Ferrari, and Luc J. Van Gool. Video mining with frequent itemset configurations. In *FIMI*, pages 360–369, 2006. URL http://dx.doi.org/10.1007/11788034_37.

[143] Basura Fernando and Tinne Tuytelaars. Mining multiple queries for image retrieval: On-the-fly learning of an object-specific mid-level representation. In *ICCV*, pages 2544–2551, 2013. URL http://dx.doi.org/10.1109/ICCV.2013.316.

[144] Winn Voravuthikunchai, Bruno Crémilleux, and Frédéric Jurie. Image re-ranking based on statistics of frequent patterns. In *ICMR*, pages 129–136, 2014. ISBN 978-1-4503-2782-4. URL http://doi.acm.org/10.1145/2578726.2578743.

[145] Li Fei-Fei. Local descriptors, sift and single object recognition. URL http://vision.stanford.edu/teaching/cs231a_autumn1112/lecture/lecture12_SIFT_single_obj_recog_cs231a_marked.pdf.

[146] O. Ore. *Galois connections*. 1944.

[147] Baptiste Jeudy and François Rioult. *Knowledge Discovery in Inductive Databases: Third International Workshop, KDID 2004, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers*, chapter Database Transposition for Constrained (Closed) Pattern Mining, pages 89–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31841-5. doi: 10.1007/978-3-540-31841-5_6. URL http://dx.doi.org/10.1007/978-3-540-31841-5_6.

[148] S. Joshi and R.C. Jain. A dynamic approach for frequent pattern mining using transposition of database. In *Communication Software and Networks, 2010. ICCSN '10. Second International Conference on*, pages 498–501, Feb 2010. doi: 10.1109/ICCSN.2010.15. URL http://dx.doi.org/10.1109/ICCSN.2010.15.

[149] François Rioult, Jean-François Boulicaut, Bruno Crémilleux, and Jérémy Besson. Using transposition for pattern discovery from microarray data. In *DMKD*, pages 73–79, 2003. URL http://doi.acm.org/10.1145/882082.882099.

[150] François Rioult. Mining strong emerging patterns in wide sage data. In *ECML/PKDD Discovery Challenge Workshop*, pages 484–487, 2004.

[151] Florent Domenach and Masato Koda. Mining association rules using lattice theory (6th workshop on stochastic numerics). pages 122–133, 2004. URL http://www.kurims.kyoto-u.ac.jp/~kyodo/kokyuroku/contents/pdf/1351-11.pdf.

[152] Laboratoire LARIM. Lattice miner. URL http://sourceforge.net/projects/lattice-miner/.

[153] VMware. Vmware virtualization for desktop & server, application, public & hybrid clouds. URL http://www.vmware.com/.

[154] Orable. Oracle vm virtualbox. URL https://www.virtualbox.org/.

Index