IEICE
TRANSACTIONS

on Information and Systems

VOL. E99-D NO. 2
FEBRUARY 2016

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

Query Bootstrapping: A Visual Mining Based Query Expansion

Siriwat KASAMWATTANAROTE[†,††a)], *Nonmember*, **Yusuke UCHIDA**[†††,††††b)], *Member*,
and **Shin'ichi SATOH**[††,††††c)], *Senior Member*

SUMMARY Bag of Visual Words (BoVW) is an effective framework for image retrieval. Query expansion (QE) further boosts retrieval performance by refining a query with relevant visual words found from the geometric consistency check between the query image and highly ranked retrieved images obtained from the first round of retrieval. Since QE checks the pairwise consistency between query and highly ranked images, its performance may deteriorate when there are slight degradations in the query image. We propose *Query Bootstrapping* as a variant of QE to circumvent this problem by using the consistency of 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.

key words: *image retrieval, instance search, query expansion, frequent itemset mining, visual word mining, query bootstrapping, adaptive support, adaptive inlier threshold*

1. Introduction

The Bag of Visual Words (BoVW) framework has been an effective means of object-based image retrieval since it was first described in Video Google [1]. It represents images as histograms of local descriptors assigned as vocabulary, namely, visual words. 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 in text retrieval [2], have been proposed in order to reduce the effect of irrelevant visual words.

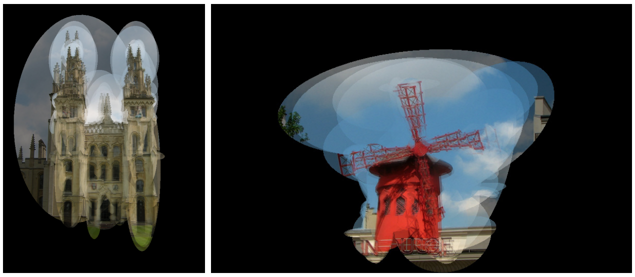


Fig. 1 The illustration of a second-round query representing an object mined from multiple relevant images by using our method.

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 used to boost the retrieval performance by selecting visual words that are geometrically consistent with the query image and the highly ranked retrieved images in the ranked list. After that, a second round of retrieval is performed. QE refines the query with selected relevant visual words such that the ranked list of the second round usually has a higher recall than that of the first round. Salton *et al.* [3] devised a simple yet effective QE, called pseudo relevance feedback or blind relevance feedback, which assumes that the top k of the retrieved ranked items are relevant and generate 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 take advantage of geometric topology information, Total Recall [4] adapts QE for 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. The formulation of the next-round query is done by averaging the BoVW histograms of the original query and the verified images with

Manuscript received May 19, 2015.
Manuscript revised September 21, 2015.
Manuscript publicized November 10, 2015.

[†] The author is with SOKENDAI (The Graduate University for Advanced Studies), Hayama-shi, 240–0193 Japan.
[††] The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
[†††] The author is with KDDI R&D Laboratories, Inc., Fujimino-shi, 356–0003 Japan.
[††††] The authors are with The University of Tokyo, Tokyo, 113–8656 Japan.
a) E-mail: siriwat@nii.ac.jp
b) E-mail: ys-uchida@kddilabs.jp
c) E-mail: satoh@nii.ac.jp

DOI: 10.1587/transinf.2015EDP7193

selected relevant visual words. This sort of refining with selected relevant visual words achieves higher recall compared with that of the first-round ranked list. 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.

To solve this problem, 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 reduces the over-dependency on the 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) to efficiently find co-occurring visual words in highly ranked images, and we also use LO-RANSAC [5] 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 Fig. 1, 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 benchmark datasets (Oxford 5k, Oxford 105k, and Paris 6k) and found that it outperforms a BoVW baseline, yields a significant performance improvement over AQE, and preserves higher robustness to query degradations.

This paper is organized as follows. Section 2 gives a brief review of related work, while Sect. 3 describes the notation used. The details of our approach are explained in Sect. 4, which consists of subsections on visual-based mining (Sect. 4.1), the adaptive *support* parameter (Sect. 4.2), integrating of the spatial verification (Sect. 4.3) and mining results and its integration to BoVW (Sect. 4.4), and speeding up the mining with the matrix transposition technique (Sect. 4.5). An evaluation of our method comparing it with the state-of-the-art is presented in Sect. 5. It shows the impact of using the top-*k* parameters, compares fixed and auto-parameter settings (Sect. 5.3), assesses the algorithm's robustness (Sect. 5.4), and compares its execution time with those of other methods (Sect. 5.5). The paper concludes in Sect. 6.

2. Related Work

Query expansion (QE) is a well-known technique in the field of text-based information retrieval. It reformulates a new query from information found in highly ranked documents. QE improves the total retrieval recall by adding terms that were not in the original query but might be in the retrieved documents. QE is regarded as a technique to improve the total retrieval recall by additional terms which were not found in the original query but might be found in the retrieved documents. 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 blind relevance feedback [3], which regards highly ranked documents as relevant. The method feeds these relevant documents back to the system 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 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.

To circumvent the above problem, [4] takes geometric information into account in QE; namely, they make a "spatial" verification (using RANSAC or the like) between the query image (or query region) and each of the database images. 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. This method lies at the heart of recent state-of-the-art visual QE methods. It imposes pairwise spatial constraints on the query image and each of the database images, 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).

Agrawal *et al.* [6] proposed frequent itemset mining (FIM) as a fast and principled method in data mining. FIM tries to find regularities, for example, in data on the shopping behavior of supermarket customers or by finding sets of products that are frequently bought together. Well-known FIM algorithms include FP-growth [7], which achieves logarithmic computational complexity against the number of

Table 1 (left) Input simple transactions of top-five images. (right) Corresponding output patterns found with *minsup* of 10% (see Sect. 4.1).

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%

transactions but requires exponential memory complexity, and LCM [8], which runs in linear computational complexity along with linear memory complexity. FIM tools are pervasively used for finding patterns in transactions. However, it is not necessary to find all item sets. FIM requires a parameter, namely the *minsup* threshold, for ensuring only the item sets that have a relatively larger fraction of transactions are output. In addition, [9] proposed an optional maximum constraint *maxsup* for judging multiple items with different criteria.

FIM has begun to be applied to problems involving images. Here, visual word mining attempts to find meaningful co-occurring patterns by taking advantage of the power of FIM especially in the context of QE, and it has been used in video mining [10], visual phrase mining [11], mining of multiple queries [12], and mining for re-ranking and classification [13]. FIM is also reported to be able to find unambiguous spatially visual meanings [11]. 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 Sect. 4.2).

3. Preliminaries

This section explains the notation used in the paper and the previous approaches, namely, BoVW, QE, AQE, and FIM.

Object-based image retrieval

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 $\boldsymbol{D} = \{I_1, I_2, \ldots, I_M\}$. In some cases, the query is a region of interest (ROI) in an image.

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. Here, Eq. 1 describes the result of such a simple retrieval scheme $\boldsymbol{R}$ for a (normally ranked) list of images.

$$\boldsymbol{R} = \{I_b \in D | I_b \text{ contains object appeared on } \boldsymbol{Q}\} \qquad (1)$$

The goal of object-based image retrieval, 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 [14].

Baseline method: BoVW

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 [15] and RootSIFT [16], [17] to obtain on average 1,000 to 3,000 descriptors per image, depending on the content of the image. We used AKM [18] to generate a large visual vocabulary (K = 1 million) and ANN [19] 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:

$$sim(\boldsymbol{Q}, I) = 1 - \left\| \frac{\boldsymbol{Q}}{\|\boldsymbol{Q}\|_1} - \frac{I}{\|I\|_1} \right\|_1 \qquad (2)$$

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.

Query expansion (QE)

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

$$\boldsymbol{Q}' = \frac{\sum_{b=1}^{k} \boldsymbol{R}_b}{k} \qquad (3)$$

where $\boldsymbol{R}_b$ denotes the BoVW histogram of the b-th image in $\boldsymbol{R}$. Some of the retrieved images may be irrelevant to the query. To circumvent this problem, AQE [4] checks the geometric consistency of each of the returned images with the 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:

$$\boldsymbol{Q}'' = \frac{\boldsymbol{Q} + \sum_{b=1}^{k'} \boldsymbol{R}_b}{k' + 1}. \qquad (4)$$

An algorithm like RANSAC [5] is typically used for the geometric consistency check. 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 Q''. In obtaining the average (Eq. 3) for the next round

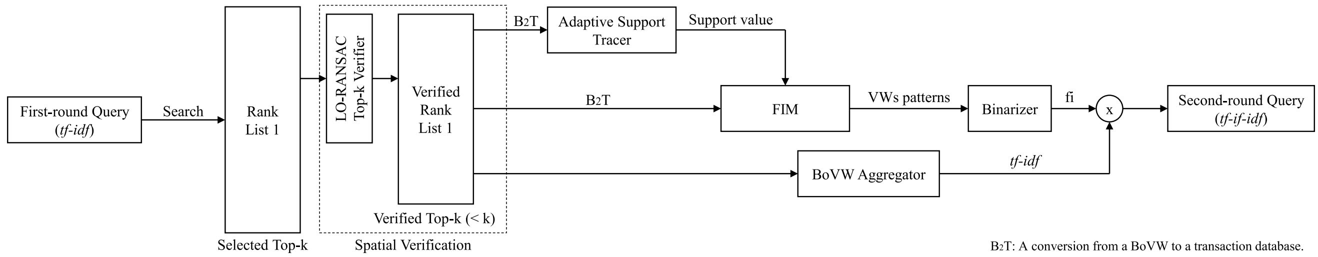


Fig. 2 Framework of Query Bootstrapping (QB) with an optional spatial verification module (SP).

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.

Frequent item sets mining (FIM)

FIM is aiming at finding regularities in transactions. 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 is also a set of items. Each pattern has a corresponding *support*:

$$\begin{aligned} sup(p, T) &= \frac{|\{t \in T | p \subseteq t\}|}{|T|} \\ s &= sup(p, T) \\ P &= FIM(T, s) \end{aligned} \tag{5}$$

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 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 [20], and it may degrade the performance of FIM. Simple outputs of FIM are shown in Table 1.

4. Query Bootstrapping

Figure 2 is an overview of our QB and QB + SP (QB combined with optional spatial verification) framework. From left to right, it works as follows:

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.

4.1 Mining Visual Transactions

In order to apply FIM to a ranked list of images, we convert the list into a set of transactions. Each image in the list is regarded as a transaction, the quantized visual words as items, and the set of images in the list as a set of transactions (we call this conversion as the B2T procedure). Given such a set of transactions, FIM outputs frequent patterns corresponding to frequently co-occurring visual words.

4.2 Adaptive *support* Parameter

FIM generates patterns under specific constraints on the *support*. On the other hand, as [9] suggested, 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. 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.

We can consider the number of patterns with intervals for support values. We assume that the number of patterns in excess of the support value has a unimodal distribution because the target object for each query is unique. The Fig. 3 shows examples that supports this assumption. Accordingly, we can determine *minsup* and *maxsup* to be the support values corresponding to the maximum number of patterns.

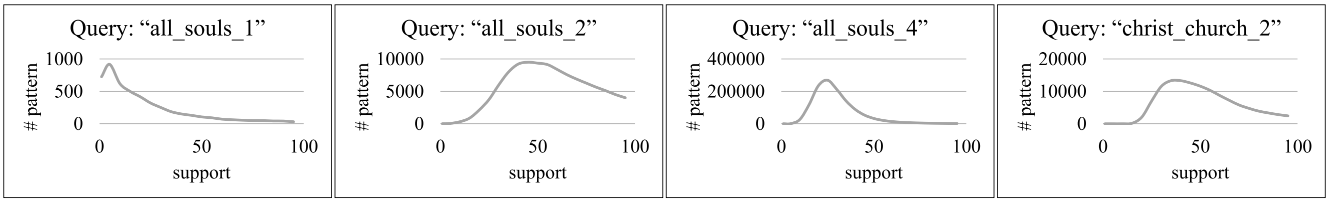


Fig. 3 Plot of the number of patterns versus support reflecting the monotonicity property of the FIM principle for four different queries. The optimal *support* parameter is at the maximum number of patterns.

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. In addition, FIM has two operation modes. In one mode, it finds closed frequent itemsets which include all patterns greater than or equal to *minsup*. In the other mode, it finds maximal frequent itemsets, which are those of its immediate supersets are frequent. 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. 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. The pseudo code of the algorithm is as follows:

```
Algorithm 1 Adaptive Support Tuning Algorithm (ASUP)
Require: $T \leftarrow B2T(\boldsymbol{R})$
procedure SUPPORT TRACER
    $b\_sup \leftarrow 0,\ n\_sup \leftarrow 100$
loop: $b\_sup < n\_sup$
    $s \leftarrow b\_sup$ ;(s as minsup)
    $S \leftarrow s + 5$ ;(S as maxsup)
    $P\_count[s] \leftarrow \|FIM(T, s, S)\|$
    $b\_sup \leftarrow b\_sup + 5$
procedure OPTIMAL SUPPORT SELECTION
    $opt\_s \leftarrow P\_count_{max}.idx \times 5$
    $opt\_S \leftarrow opt\_s + 5$
return $opt\_s, opt\_S$
```

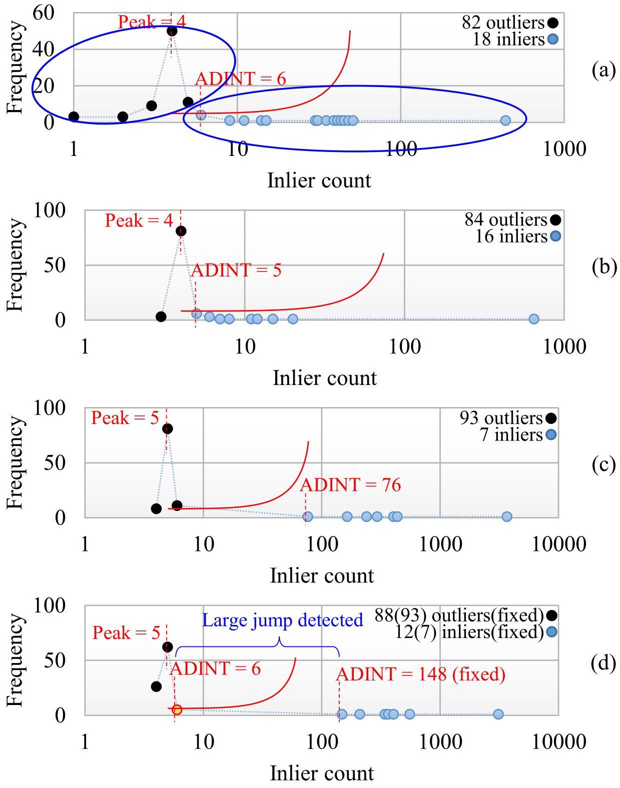


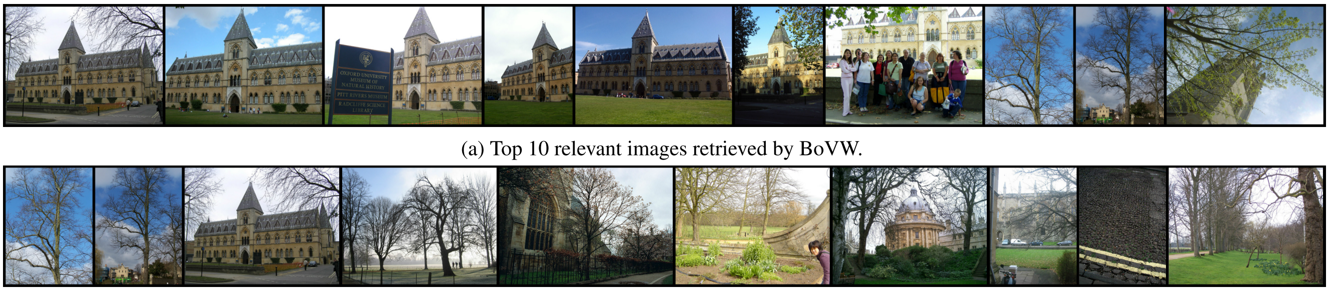
Fig. 4 The actual cases of how ADINT finds an inlier threshold. The dots represents the frequency of each inlier count (shown in log scale). The ADINT ratio was set to 0.9 (red arc). (a) Blue ellipses show the group of outlier images (left) and inlier images (right). (b,c) General cases. (d) Special case when ADINT finds a threshold at the boundary of the outlier group (red-circle dot), which is then fixed to the inlier group.

Finally FIM is run in closed frequent itemsets mode with the optimal *minsup* and *maxsup* in order to generate actual patterns. Note that using *minsup* and *maxsup* helps to improve the quality of the generated patterns on an exact frequent object and hence contributes to the overall retrieval performance, and the closed frequent itemsets mode is faster in this case since there is no need to discover unnecessary patterns outside the min/max support boundary.

This step assumes that the first round retrieval results are dominated by relevant images containing target objects. Unfortunately, as Fig. 5 implies, this is not always the case. In other words, FIM and the parameter determination algorithm explained here may not always work so well. To cope with this problem, we can optionally employ spatial verification.

4.3 Improving QB with LO-RANSAC

We use LO-RANSAC for the spatial verification. LO-RANSAC generates a verified ranked list from a query image and the images in the ranked list. It finds the maximum number of point pairs between given two images that are geometrically consistent in some way (we use Homography). Such point pairs are called inliers. Images having more inliers than a threshold are called verified images. Too tight of a threshold may generate very few (or no) verified images, while too loose a threshold is equivalent to no verification.



(a) Top 10 relevant images retrieved by BoVW.

(b) Top 10 relevant images retrieved by QB.

Fig. 5 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 (SP) process, QB produces a result closer to the dominated one.

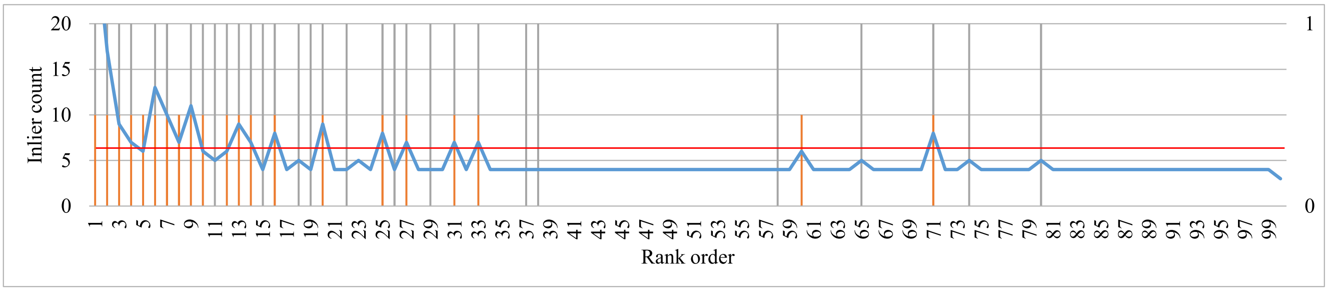


Fig. 6 The blue line shows the total number of inliers found in the top 100 images of the initial rank on the sample query with a baseline AP of 30.14%. The gray bar shows the true positive images appearing in the ranked list, whereas the orange bar shows images selected by our adaptive inlier threshold (ADINT) seen in a red line that yielded a final AP of 82.07%.

Adaptive Inlier Threshold

We assume that images having target objects tend to have large numbers of inliers, while images without target objects tend to have few inliers. 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 Fig. 4 (a)–(b). Generally, images with high inlier counts tend to be the correct matches (see the right ellipse in Fig. 4 (a)), while images with low inlier counts tend to be outliers (see the left ellipse in Fig. 4 (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:

In order to determine the inlier threshold between inlier and outlier images, the algorithm 2 firstly finds a point in a histogram where the most images are belonging to. This 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. The ratio is introduced, namely, ADINT ratio, with a typical value of 0.9 for controlling the radius.

Algorithm 2 Adaptive Inlier Threshold (ADINT)

```
Require: $inls$ (as inlier count distribution)
procedure ADINT
    $center \leftarrow find\_max(inls)$
    $adint_{ratio} \leftarrow predefined$
    $inl \leftarrow inls[center * adint_{ratio}]$
loop: $center.idx < inl.idx$
    $a \leftarrow center - inl$
    $b \leftarrow |center.idx - inl.idx|$
    $c \leftarrow \sqrt{a^2 + b^2}$
    if $center * adint_{ratio} \leq c$ then
        $adint = inl$
        if $find\_large\_jump\_ahead(inls, adint) \neq null$ then
            $adint = find\_large\_jump\_ahead(inls, adint)$
        stop;
    $inl \leftarrow inls[inl.idx - 1]$
return $adint$
```

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 Fig. 4 (c) and (d)). On the other hands, setting an ADINT ratio to 0.9 might not be enough, and may fail to filter out some irrelevant images as seen in Fig. 4 (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 6 compares the auto-selected inlier images (orange bars) with the ground truth (gray bars).

4.4 Integrating QB into the BoVW Framework

Up to now, in order to incorporate the set of patterns obtained by FIM, we extend the *tf-idf* weight for each quantized visual word by taking into account the frequent item (*fi*) that processed by the prior steps as a representative form of the occurrences of visual words. In this work, *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*, is easily defined as follows:

$$\textit{tf-fi-idf} = \textit{fi} \times \textit{tf-idf} \qquad (6)$$

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

4.5 Speed Up Mining Process with Transaction Transposition

Given transactions with m items, FIM normally discovers shared patterns from 2^m possible subsets of combinations of items in a set of transactions. As a toy example, suppose we have three code words a, b, and c; the total number of possible patterns will be $2^3 = 8$, and the complete set of patterns will be $P \in \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$.

Since we use 1 million codewords in the BoVW framework, the resultant BoVW vectors are very high dimension, and their 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 × 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 make 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 [21]–[23] have tried to reduce a dimension of m, by using a Galois connection to do mapping on a completed lattice from a transposed matrix side back to the original untransposed matrix side[†].

Transposing a transaction database, which contains m items with n transactions where $2^m >> 2^n$, enables FIM to operate on a much smaller space of patterns. This means FIM can work much faster on the transpose than on the original transactions (see Fig. 7). Table 2 compares time used per query of FIM on a transaction database and its transpose.

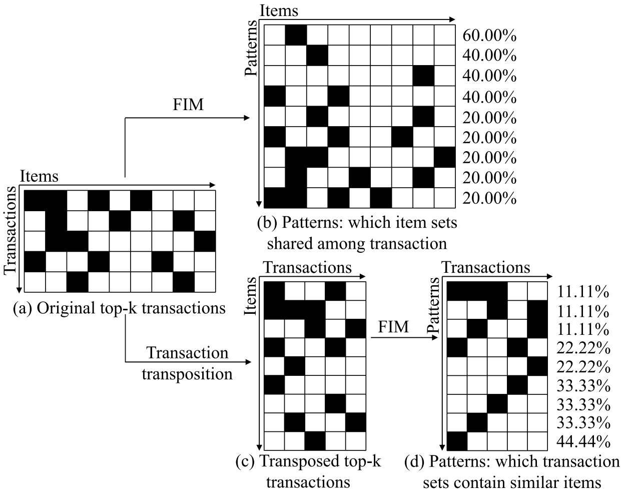

Items
Transactions
(a) Original top-k transactions
FIM
Items
Patterns
60.00%
40.00%
40.00%
40.00%
20.00%
20.00%
20.00%
20.00%
20.00%
(b) Patterns: which item sets shared among transaction
Transaction transposition
Transactions
Items
(c) Transposed top-k transactions
FIM
Transactions
Patterns
11.11%
11.11%
11.11%
22.22%
22.22%
33.33%
33.33%
33.33%
44.44%
(d) Patterns: which transaction sets contain similar items

Fig. 7 (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.

Table 2 Average time usage per query when using FIM on a database and on matrix transposition of a database (FIMT).

	Time (second)		
	Ox 5k	**Ox 105k**	**Paris 6k**
FIM	7.857	0.274	Upper limit
FIMT	1.274	0.038	12.132

5. Evaluation and Comparison

We compared BoVW ([24]–[26]), average query expansion (AQE) [4], query bootstrapping (QB), and QB with spatial verification (QB + SP). To seek for fair evaluation, we used three state-of-the-art datasets and their ground truth labels, namely, Oxford 5k, Oxford 105k, and Paris 6k [18], [27], which were largely used in the object based image retrieval works.

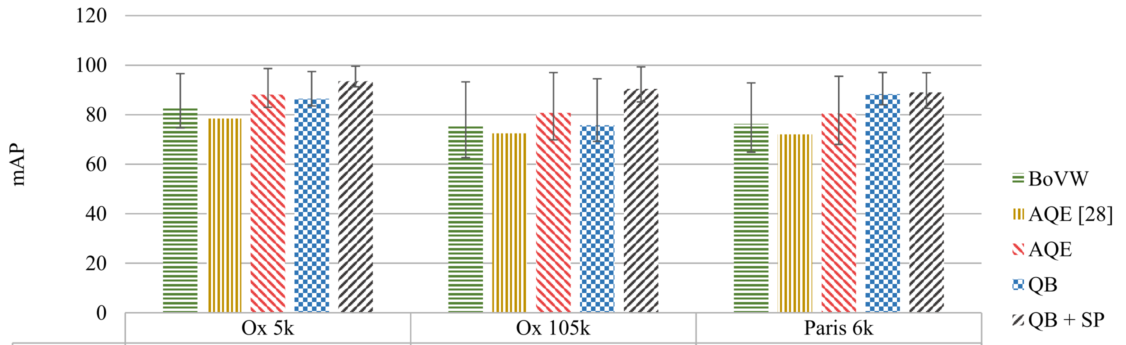
The Oxford Buildings 5k dataset consists of 5062 high-resolution images that were crawled from Flickr by using queries for famous Oxford landmarks, such as “Oxford Christ Church” and “Oxford Radcliffe Camera”.

The Oxford Buildings 105k dataset is an extension of Oxford 5k that uses the distractor images of the Flickr 100k dataset, which consists of 100071 images collected by searching for popular Flickr tags.

The Paris 6k dataset consists of 6412 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 is provided differently on Oxford and Paris dataset. Each of which are 11 different land-

[†] For details on how to transpose a visual transaction and how to map back mining results by using Galois connections, we encourage readers to access the URL: http://www.satoh-lab.nii.ac.jp/%7estylix/papers/siriwat%5fqbtranspose.pdf



	Ox 5k	Ox 105k	Paris 6k
≡ BoVW	82.84	75.66	76.33
⦀ AQE [28]	78.50	72.50	72.00
▧ AQE	88.12	80.71	80.44
▦ QB	86.41	75.67	88.28
▨ QB + SP	**93.49**	**90.36**	**88.96**

Fig. 8 Comparison of our proposed methods (*QB* and *QB + SP*) with BoVW, AQE, and AQE [28] on Oxford 5k, Oxford 105k, and Paris 6k. The bars show QB and QB + SP significantly outperformed the other methods. Also, the 1st and the 3rd quartile bars show our methods also performed lower variances among queries than the others.

marks, each consisted of 5 possible queries. This gives a set of 55 queries 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. *Junk* — less than 25% of the object is visible, or there is a very high level of occlusion or distortion.
4. *Bad* — the object is not present.

In term of evaluation, we used mean average precision (mAP) as the evaluation metric. In addition, we regard images with *Good* and *OK* labels as relevant, images with *Bad* label as irrelevant, and we disregard images with *Junk* label in the calculation.

We used 100 million randomly sampled local features to train a codebook of 1 million code words. In particular, we used an approximate nearest neighbor search (Fast-Cluster [18]) with 50 iterations. The visual word assignment was done using fast library for approximate nearest neighbor (FLANN [19]) with a hard assignment, 512 checks on maximum leafs, and the KDTree index. For the spatial verification, we used the default RANSAC parameters, while we increased the random samples from 1000 to 3000 in the hope of obtaining a small accuracy improvement. Moreover, as indicated above, we enabled the local optimization option, *i.e.*, LO-RANSAC.

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 and 512 GBs of memory. The core retrieval module was written in the C++11 standard and compiled with the optimization -O3 flag using cross-compiled GCC 4.8.1. We evaluated the time consumption of FIM and similarity scoring based on a single CPU. Since we parallelized RANSAC, the times reported in this paper are for all of the available CPUs.

5.1 Overall Evaluation

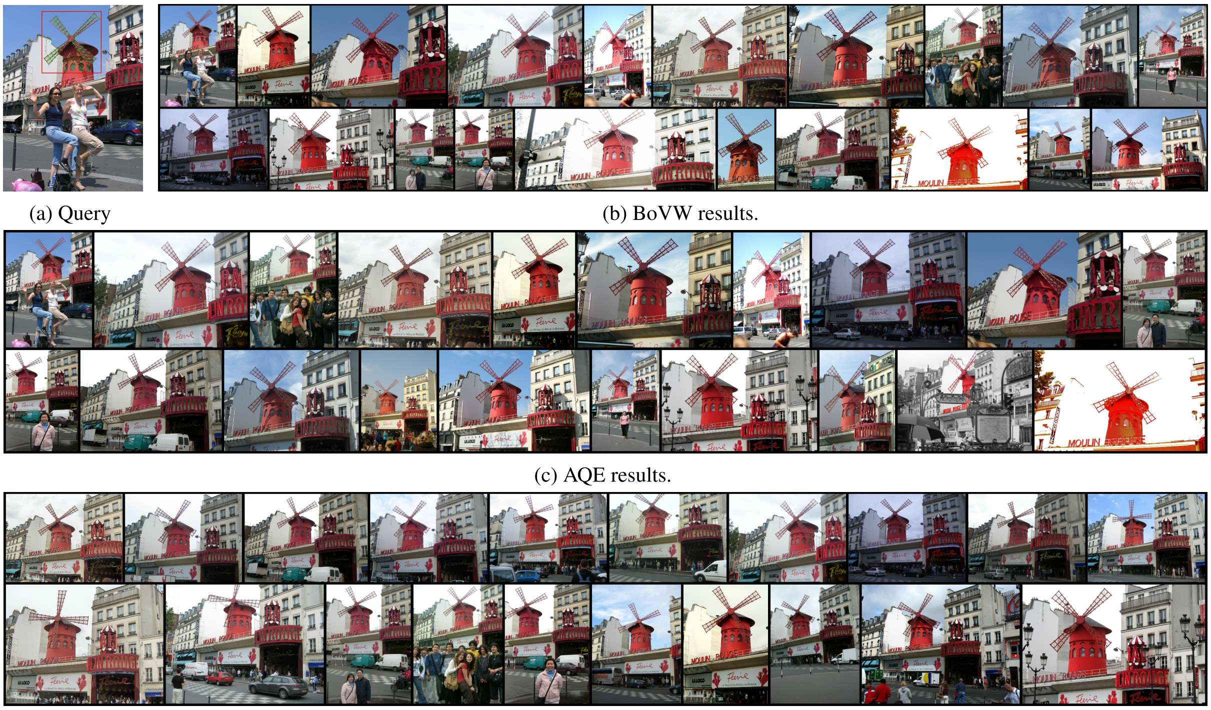
Figure 8 shows the performance of BoVW, AQE, AQE [28], QB, and QB + SP. The evaluation was done based on the same parameter configuration for all approaches. However, for the QE based approaches (AQE, AQE [28], QB, and QB + SP), setting various top-*k* values will directly effect to the final performance. 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 as reported in the Sect. 5.2. However, for the AQE [28], due to a speed issue, top-10 images is only the maximum for them.

It is clear from this figure that QB + SP significantly outperformed the other methods with the lower variances among queries. Figure 9 shows samples of the rank lists retrieved with the BoVW, AQE, and QB + SP.

5.2 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 Fig. 10 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



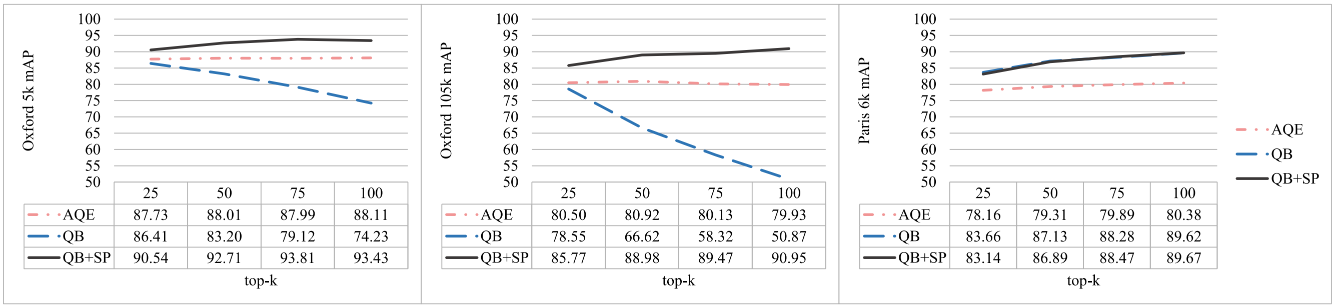
(a) Query

(b) BoVW results.

(c) AQE results.

(d) QB + SP results.

Fig. 9 Top 20 relevant images showing how the three main approaches differ in effect. (a) Query image with ROI (b) BoVW result (c) AQE got better matches to ROI than BoVW (d) QB + SP reflects the frequent objects from the initial rank.



Oxford 5k mAP	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

Oxford 105k mAP	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

Paris 6k mAP	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

Fig. 10 Impact of different top-*k* values on the retrieval performance of the query expansion based method.

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 10 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.

5.3 Automatic Parameters and Relative Improvement

We evaluated the performance of the automatic parameter tuning algorithms including *support*, and *inlier threshold*.

Adaptive support

Here, we compared the performance in terms of mAP of QB + SP 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 11 shows the results. *Minsup* had a strong impact on the final performance of Oxford 5k, and the best fixed *minsup* was around 20-30 (blue line). However, the adaptive support algorithm performed better than

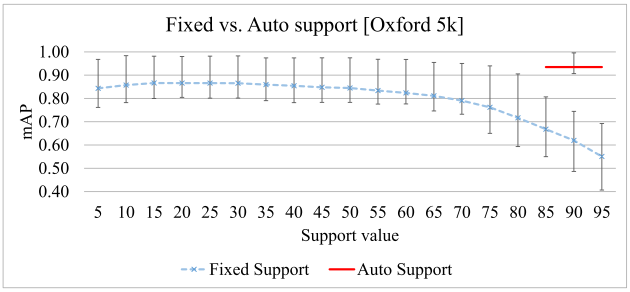


Fig. 11 Retrieval performance of QB + SP on Oxford 5k dataset comparing the impact of a fixed *support* (blue line) and adaptive *support* (red line). The 1st and the 3rd quartile bar show that the adaptive method achieves lower variances among queries.

Table 3 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.

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

the non-adaptive one, with lower variances among queries (red line).

Adaptive inlier threshold

We compared the effect of the adaptive inlier threshold algorithm (ADINT) with that of a fixed inlier threshold (FINT). For fairness, we compared only QB + SP and AQE. The results are shown in Table 3. AQE had less impact on the various inlier thresholds, while QB + SP had a strong impact. ADINT significantly improved the performance of QB + SP, while its contribution to AQE was minor.

5.4 Impact of Query Quality on Retrieval Robustness

The following experiments assessed the impact of degraded queries.

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 Fig. 12, 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.

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 13 shows that QB + SP performed the best until 40% whereas other methods experienced large performance drops.

Table 4 Average time consumption per query (without feature extraction) of BoVW, AQE, QB, and QB + SP.

	Time (second)			
	BoVW	**AQE**	**QB**	**QB + SP**
Ox5k	0.059	0.305	1.374	5.082
Ox105k	0.793	1.465	1.042	5.557
Paris6k	0.667	0.361	12.275	10.981

Table 5 Average time consumed per query for each module of QB and QB + SP.

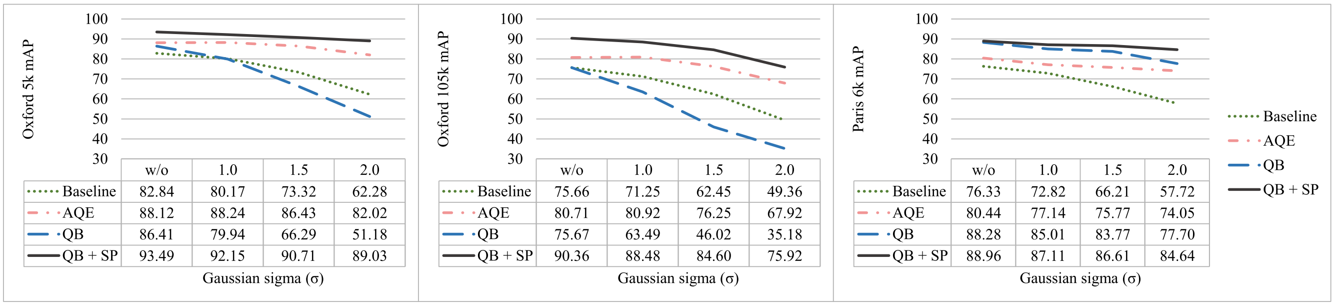
	Time (second)				
	QB		**QB + SP**		
	FIMT	**Sim**	**SP**	**FIMT**	**Sim**
Ox 5k	1.274	0.041	0.224	4.495	0.305
Ox 105k	0.038	0.211	0.243	4.398	0.124
Paris 6k	12.132	0.040	0.229	10.545	0.105

5.5 Time Consumption

Table 4 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 Table 5 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 14 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 ≥ 100k).

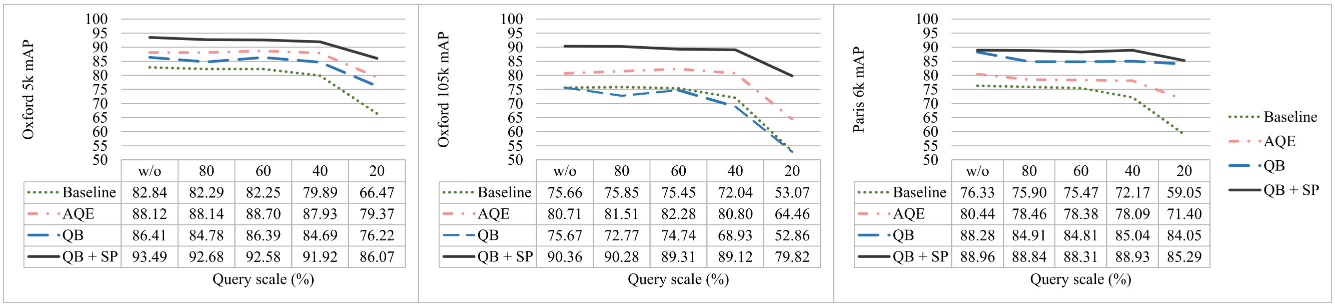


Oxford 5k mAP	w/o	1.0	1.5	2.0
Baseline	82.84	80.17	73.32	62.28
AQE	88.12	88.24	86.43	82.02
QB	86.41	79.94	66.29	51.18
QB + SP	93.49	92.15	90.71	89.03

Oxford 105k mAP	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

Paris 6k mAP	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

Fig. 12 Retrieval performance for synthetic noisy query.



Oxford 5k mAP	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

Oxford 105k mAP	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

Paris 6k mAP	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

Fig. 13 Retrieval performance for simulated low-resolution query.

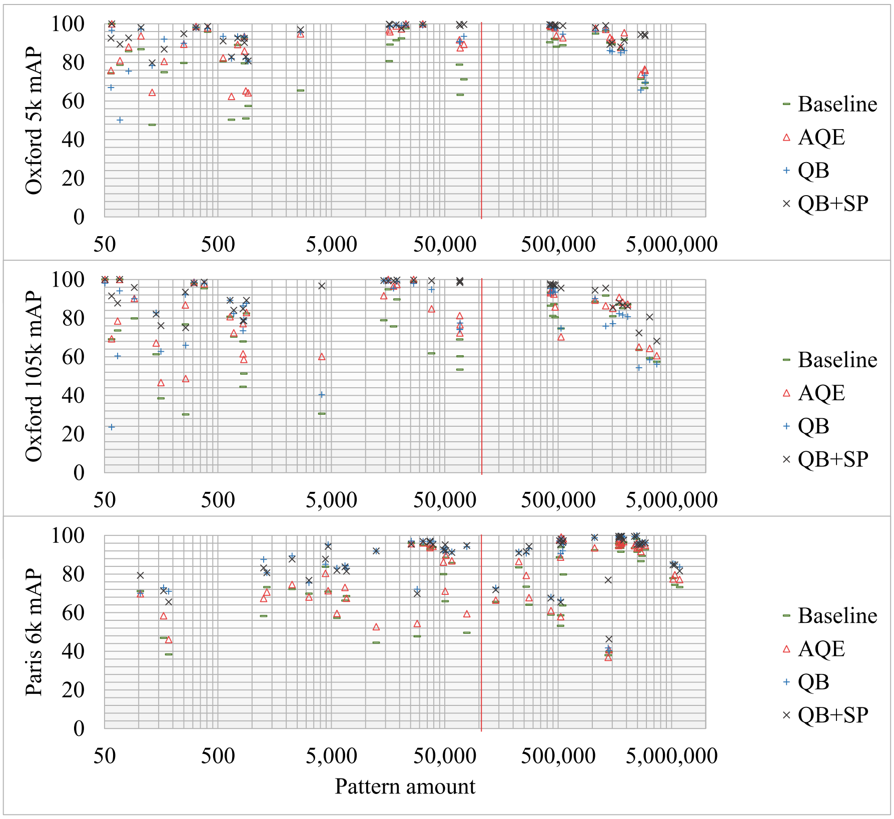


Fig. 14 mAP vs. total number of patterns. This plot shows the improvement was performed by our QB + SP on most queries that generated less than 100k patterns. In contrast, it had less effect on queries that generated more than 100k patterns.

Table 6 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.

	Type	**#Topics**	**Baseline**	**QB**				**QB+SP**			
				FIMT(s)	Precision(%)			FIMT(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**
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**

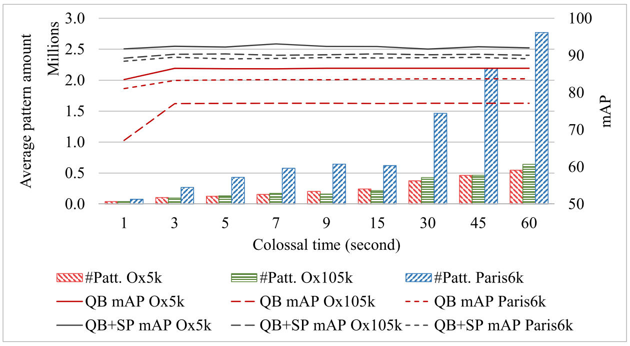


Fig. 15 Number of patterns versus time. The discovery of rest patterns yielded no big improvement in mAP after it had produced enough core patterns about 100k patterns, which were processed within 3 seconds.

The red vertical lines in Fig. 14 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 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.

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.

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 [20], 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 [29]. In our study, while mining "easy" queries could be accomplished very quickly (within a few hundred milliseconds), the "hard" cases were time consuming. According to the experimental results in Fig. 15, 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).

6. Conclusions

The paper proposed Query Bootstrapping (QB), a variant of Query Expansion (QE), to better exploit the correlation in the first-round retrievals. 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. Furthermore, we presented an on-the-fly *support* tuning algorithm for providing the best *support* values (*i.e.*, *minsup* and *maxsup*) for the visual mining process. In addition, we presented an automated *inlier threshold* algorithm for LO-RANSAC spatial verification. The patterns found by FIM are taken into account in a new weighting scheme called *tf-fi-idf*.

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. The results show our approach has reached a new level of performance far beyond that of the classic BoVW framework or even AQE.

References

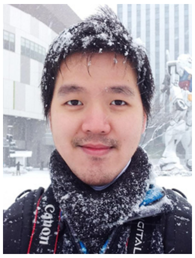
[1] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," ICCV, pp.1470–1477, 2003.

[2] R.A. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley Longman Publishing, 1999.

[3] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," Journal of the American Society for Information Science, pp.355–364, 1990.

[4] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," ICCV, pp.1–8, 2007.

[5] K. Lebeda, J. Matas, and O. Chum, "Fixing the locally optimized RANSAC," BMVC, pp.95.1–95.11, 2012.
[6] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," VLDB, pp.487–499, 1994.
[7] C. Borgelt, "An implementation of the fp-growth algorithm," OSDM, pp.1–5, 2005.
[8] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "An efficient algorithm for enumerating closed patterns in transaction databases," Discovery Science, vol.3245, pp.16–31, 2004.
[9] Y.-C. Lee, T.-P. Hong, and W.-Y. Lin, "Mining association rules with multiple minimum supports using maximum constraints," IJAR, vol.40, no.1-2, pp.44–54, 2005.
[10] T. Quack, V. Ferrari, and L.J.V. Gool, "Video mining with frequent itemset configurations.," FIMI, vol.4071, pp.360–369, 2006.
[11] J. Yuan, Y. Wu, and M. Yang, "Discovery of collocation patterns: from visual words to visual phrases," CVPR, pp.1–8, 2007.
[12] B. Fernando and T. Tuytelaars, "Mining multiple queries for image retrieval: On-the-fly learning of an object-specific mid-level representation," ICCV, pp.2544–2551, 2013.
[13] W. Voravuthikunchai, B. Crémilleux, and F. Jurie, "Image re-ranking based on statistics of frequent patterns," ICMR, pp.129–136, 2014.
[14] O. Paul, F. Jon, S. Greg, J. David, and M. Martial, "Trecvid 2014 — an overview of the goals, tasks, data, evaluation mechanisms, and metrics," NIST, USA, 2014.
[15] M. Perdoch, O. Chum, and J. Matas, "Efficient representation of local geometry for large scale object retrieval," CVPR, pp.9–16, 2009.
[16] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol.60, no.2, pp.91–110, 2004.
[17] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," CVPR, pp.2911–2918, 2012.
[18] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," CVPR, pp.1–8, 2007.
[19] M. Muja and D.G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," VISAPP, pp.331–340, 2009.
[20] F. Zhu, X. Yan, J. Han, P.S. Yu, and H. Cheng, "Mining colossal frequent patterns by core pattern fusion," ICDE, pp.706–715, 2007.
[21] F. Rioult, J.-F. Boulicaut, B. Crémilleux, and J. Besson, "Using transposition for pattern discovery from microarray data," DMKD, pp.73–79, 2003.
[22] F. Rioult, "Mining strong emerging patterns in wide sage data," ECML/PKDD Discovery Challenge Workshop, pp.484–487, 2004.
[23] F. Domenach and M. Koda, "Mining association rules using lattice theory," 6th Workshop on Stochastic Numerics, pp.122–133, 2004.
[24] C.-Z. Zhu and S. Satoh, "Large vocabulary quantization for searching instances from videos," ICMR, pp.1–8, 2012.
[25] C.-Z. Zhu, H. Jégou, and S. Satoh, "Query-adaptive asymmetrical dissimilarities for visual object retrieval," ICCV - International Conference on Computer Vision, pp.1705–1712, 2013.
[26] C.-Z. Zhu, S. Kasamwattanarote, X. Wu, and S. Satoh, "Tell me about tv commercials of this product.," MMM, pp.242–253, 2014.
[27] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," CVPR, pp.1–8, 2008.
[28] O. Chum, A. Mikulik, M. Perdoch, and J. Matas, "Total recall II: Query expansion revisited," CVPR, pp.889–896, 2011.
[29] P.S. Yu, X. Yan, J. Han, H. Cheng, and F. Zhu, "Approximate frequent pattern mining."



Siriwat Kasamwattanarote is a PhD candidate at Sokendai, Japan, under the supervision of Prof. Shin'ichi Satoh. He received MS degree in Computer Science and Information Technology from the Chulalongkorn University, Thailand in 2012 and the BS degree in Information and Communication Technology from the Mahidol University, Thailand in 2009. His research interests include image processing, multimedia retrieval, information mining, and software optimization.

Yusuke Uchida received his Bachelor Degree of Integrated Human Studies from Kyoto University, Kyoto, Japan, in 2005. He received a degree of Master of Informatics from Graduate School of Informatics, Kyoto University, in 2007. His research interests include large-scale content-based multimedia retrieval, augmented reality, and image processing. He is currently with KDDI R&D Laboratories, Inc. and is a PhD Candidate at the University of Tokyo.

Shin'ichi Satoh is a professor at National Institute of Informatics (NII), Tokyo. He received PhD degree in 1992 at the University of Tokyo. His research interests include image processing, video content analysis and multimedia database. Currently he is leading the video processing project at NII, addressing video analysis, indexing, retrieval, and mining for broadcasted video archives.