Minimizing the Number of Keypoint Matching Queries for Object Retrieval

Johannes Niedermayer
niedermayer@dbs.ifi.lmu.de
Peer Kröger
kroeger@dbs.ifi.lmu.de

Institute for Informatics
LMU Munich
Munich, Germany

Abstract

To increase the efficiency of interest-point based object retrieval, researchers have put remarkable research efforts into improving the efficiency of *k*NN-based feature matching, pursuing to match thousands of features against a database within fractions of a second. However, due to the high-dimensional nature of image features that reduces the effectivity of index structures (curse of dimensionality) and due to the vast amount of features stored in image databases (images are often represented by up to several thousand features), this ultimate goal demanded to trade *k*NN query runtimes for query precision. In this paper we address an approach complementary to indexing in order to improve the efficiency of retrieval by querying only the most promising keypoint descriptors, as this affects *k*NN matching time linearly. As this reduction of *k*NN queries reduces the number of tentative correspondences, a loss of query precision is minimized by an additional image-level correspondence generation stage with a computational performance independent of the underlying indexing structure. Our experimental evaluation suggests good performance on a variety of datasets.

1 Introduction

While the development of the SIFT-Descriptor [24] made effective object retrieval on a large scale feasible, its initial use of nearest neighbor queries lead to slow runtimes even on relatively small data sets. These slow runtimes were first compensated by rough quantization using the Bag of Visual Words (BoVW) technique [36]. In recent years, the focus turned back more and more to approximate *k*NN queries [2, 8, 21, 26] due to their possible gain in matching accuracy [18]: *k*NN queries provide an accurate ranking of the match candidates and a measure of proximity between database features and query vectors. Recently, a remarkable leap in performance has been achieved concerning efficient and effective *k*NN query processing in computer vision. However, with the vast amount of features that have to be matched during recognition (up to a few thousand), even very fast *k*NN indexing techniques that can provide approximate query results in under ten milliseconds (e.g. [21]), would yield recognition runtimes of several seconds.

We argue that the use of *k*NN queries for object recognition in large-scale systems cannot be achieved by developing efficient indexing techniques alone. The problem of efficiency has to be approached from additional research directions as well, such as the *number* of *k*NN queries posed on the system. In this paper, we evaluate an alternative recognition pipeline that

Pages 48.1-48.13
DOI: https://dx.doi.org/10.5244/C.29.48

ranks query features by assessing their matchability. In order to reduce the number of kNN queries, only the most promising features in this ranking are matched against the database. However, despite gaining efficiency, the enforced reduction of kNN queries causes a reduction of feature matches, decreasing the quality of the query result. While recall can be increased by increasing k, to increase Mean Average Precision (MAP) we propose to expand matches on the image level: Given a single seed feature match in a candidate image, this match is expanded by comparing its spatially neighboring keypoints. This step pushes load from the matching step (with complexity mostly determined by the underlying index structure) to an additional step that only has to consider the features stored in a single image pair. Therefore, this work stands in contrast to research in the area of BoVW-based retrieval: Research involving the BoVW pipeline assumes that the matching step is relatively cheap, especially if approximate cluster assignment techniques [25, 28] are used. In contrast, this paper aims at maximizing MAP for a small number of processed features, assuming that feature matching is expensive. This different optimization criterion is especially of interest as techniques that do not lead to significant gains in performance at a high number of features (where convergence to the maximum possible MAP has already been achieved by other techniques) can lead to a remarkably higher MAP when only a low number of features is queried. Given this argumentation, the contribution of this paper is to provide a *simple* and *extensible* pipeline for *large-scale* object retrieval based on *kNN queries* with *all* of the following properties:

- *Reduction of the number of keypoints queried* by using a *general* keypoint ranking scheme in order to reduce matching times.
- Acceleration of the pipeline by *state-of-the art index structures* such as (Locally Optimized) Product quantization [17] or Multi-Index-Hashing [26].
- *Geometric Match Expansion* to relieve the index structure and to increase query *MAP*.
- The use of *kNN queries* ($k >> 2$) to increase the number of seed hypotheses and *recall*.
- Consideration of feature *distances* during scoring to allow accurate scoring of features.

We also provide an evaluation of this pipeline on a variety of datasets. We evaluate the effect of k in relation to the number of keypoints queried on the systems performance, and the pipeline's behaviour on different feature descriptors including real-valued and binary features.

This paper is organized as follows. Section 2 formally defines the problem addressed in this paper. We then review related work in Section 3. In Section 4 we describe our solution to reducing the number of kNN queries during retrieval. Section 5 evaluates our solution on different feature types and datasets. Section 6 concludes this work.

2 Problem Definition

Accurate kNN queries are, even after astonishing research efforts in the last years, still computationally expensive. Therefore, in addition to indexing efficiency, other possibilities must be considered to reduce the complexity of feature matching. Generally, to achieve this complexity reduction, different approaches are reasonable, such as dimensionality reduction, reducing the cost of distance functions (e.g. Hamming distance on binary features), or decreasing the number of matching queries.

In this paper we focus on the latter approach: Let a database of images, represented by sets of features describing the neighborhood around interest points, be given. Let n denote the upper bound on the number of matching queries, constraining the number of kNN queries. The goal of this research is to develop a retrieval algorithm that returns a list of images ranked by their visual similarity to the query. We aim at modifying the image recognition pipeline such that a given performance measure (in our case MAP) is maximized for a given n.

The problem setting is similar to BoVW-based approaches, however in such a context it is usually assumed that $n = n_{max}$. In this paper we address the opposite case where $n << n_{max}$.

3 Related Work

Keypoint Reduction. In order to reduce the number of extracted features that have to be matched, [11] aimed at predicting the matchability of features by interpreting the problem as a classification task. Keypoint reduction can also be achieved by employing the Adaptive Non-Maxima suppression (ANMS) from Brown et al. [3]. Their approach aims at finding interest points that are sufficiently distributed across the whole image and is computationally relatively inexpensive. Hajebi and Zhang [10] propose to keep track of the distribution of scores during query processing and stop the investigation of further features as soon as the score difference between the best-scored image and the average score becomes large enough. Other approaches to rank features are based on visual attention [22]. The approaches from Sattler et al. [32, 33], based on BoVW, consider features in ascending order of the length of inverted lists of the corresponding visual words. While we aim at reducing the number of query features with feature ranking, there exist also approaches aiming at reducing the number of database features [23, 37, 40]. Decreasing both database and query features can be a useful choice in the context of our pipeline, but is beyond the scope of this paper.

***k*NN Indexing.** Due to the curse of dimensionality, exact *k*NN query processing in high-dimensional space often suffers from slow runtimes. Therefore indexing research in the image community concentrates on *approximate* nearest neighbor search. Recent research on *k*NN indexing aims at providing low runtime and storage complexity while providing accurate distance approximations at the same time. One group of these techniques is based on the Product Quantization approach from Jégou et al. [17]. Recent extensions of this approach include [2, 8, 21]. Another group of techniques aiming at efficient query processing is built on the idea of generating distance-preserving binary codes from real-valued features, sometimes referred to as *binarization* [12, 13, 19, 38, 42]. In contrast to binarization techniques, binary keypoint descriptors such as BinBoost and ORB [31, 39] can avoid the indirection of extracting real-valued (e.g. SIFT) features first and then binarizing them. Nearest Neighbor queries on databases of binary features can be sped up using LSH [14] or Multi-Index Hashing [26].

***k*NN-based Matching.** *k*NN-based matching has a long history in image retrieval and has been used e.g. by Lowe [24] ($k = 2$). Jégou et al. [18] evaluated *k*NN matching based on local features, especially SIFT, however without addressing keypoint reduction. They proposed a voting scheme optimized for *k*NN-based retrieval. Their adaptive criterion scores matches relative to the distance of the *k*th match. The authors also analyzed normalization methods for the resulting scores. Qin et al. [30] proposed a normalization scheme for SIFT-features that locally reweighs their Euclidean distance, optimizing the separability of matching and non-matching features.

Match Expansion. As our technique aims at reducing the number of *k*NN queries during the matching step, the generation of a sufficient number of match hypotheses has to be achieved in a different fashion. We do so by applying a flood-filling approach using *k*NN matches as seed points. Match expansion has received quite some attention in the computer vision community [6, 7, 9, 20, 34, 35, 36, 37]. One of the first techniques in this area of research has been proposed by Schmid and Mohr [35] which does however not consider feature reduction. Sivic and Zisserman adapted their technique for Video Google [36]. We however do not reject matches based on this technique but rather increase the score of a given image by considering neighboring features. Our work is also inspired by [34], where

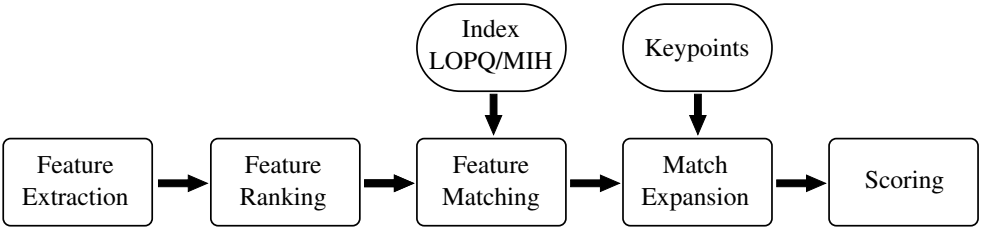

Figure 1: Recognition pipeline. The last three steps, i.e. matching, expansion, and scoring are performed tightly interleaved for each query feature.

the authors used a region-growing approach for establishing correspondences in the context of multi-view matching in order to increase result quality, but not to reduce the number of queried features. The approach from Ferrari et al. [7] builds a dense grid of features over the image while we use the initially provided keypoints and descriptors that are stored in the database nonetheless, reducing computational overhead. Geometric min-Hashing [5] aims at increasing precision at the cost of recall, by dropping features that do not share a similar neighborhood. However, if we reduce the number of matching queries, one of the main concerns is recall, such that our approach aims at increasing MAP without negatively affecting recall. The authors of [33] combined keypoint reduction and concepts similar to our match expansion, however in the context of 2D-to-3D matching and pose estimation, employing Lowe's SIFT ratio test without kNN-based scoring.

4 Pipeline

The general retrieval pipeline from this paper follows the one used in the past for BoVW-based image retrieval, but in order to incorporate kNN queries and to reduce the number of query features we had to apply some changes, see Figure 1. The pipeline was designed with extensibility in mind such that each stage, e.g. keypoint reduction and match expansion, can be easily exchanged by different techniques. Furthermore, it can be extended with geometric verification or query expansion techniques [4].

1) Feature Extraction. During feature extraction, given the query image, we extract the set of keypoints and descriptors. Possible features include floating point features such as SIFT [24] or binary features such as BinBoost and ORB [31, 39]. The number of features depends on the feature extractor and can range up to several thousand features.

2) Feature Ranking. Feature ranking is based on the idea that some features in an image contain more information than other (such as vegetation) features. We aim at ordering the extracted features by a given quality measure and query only the features with the highest chance of providing good match hypotheses. There exist several techniques for feature ranking, the only criterion such a technique needs to fulfill in order to be integrated into the recognition pipeline is that it returns a quality score for each query feature. Besides simple baselines such as random ranking or ranking by response or size, sophisticated techniques include Adaptive Non-Maximal Suppression [3] and the use of decision trees [11]. The result of this feature ranking step is a list of features ordered by their expected matching quality.

3) Feature Matching. Feature matching aims at finding match hypotheses for the highest ranked features. For each of the first n features in the ranking, a kNN query is posed on the database. The selection of the parameter k of the kNN query is important for maximizing the quality of the query result [18], as a large k introduces a high number of erroneous correspondences. A small k however reduces the retrieval quality, as many high-quality

hypotheses are left unconsidered. Basically, k can be seen as a way to tweak *recall* at a given number of query features, as the number of images returned by the query is at most $n * k$. Therefore, if a very small number of kNN queries is used for correspondence generation, it is possible that a very large k increases effectiveness, as it allows for finding more initial correspondences. We refer to Section 5 for an experimental analysis of this problem. The feature matching stage provides a list of tentative feature matches (p_q^i, p_x^j). To increase the efficiency of feature matching, we rely on fast state-of-the-art (approximate) indexing techniques optimized for high-dimensional data, focusing on Locally Optimized Product Quantization [21] for real-valued features and Multi-Index Hashing [26] for binary features.

4) Match expansion. The match expansion phase is tightly interleaved with the feature matching phase. In our scenario where we want to pose a very small number of kNN queries on the system, we face the problem that even if we find some correspondences between the query and a database image, their number will be relatively low, increasing the probability that a good match is outranked by an image containing common random matches only. Therefore, we shift the load of correspondence generation from the matching stage –that employs kNN queries– to an intermediate stage that avoids such queries. Match expansion aims at reducing the runtimes of generating additional matches, which usually depend on the underlying index structure, to runtimes depending on the features stored in a single image pair. When employing exhaustive search with product quantization for indexing, match expansion therefore avoids additional linear scans over the feature database; as non-exhaustive variants of product quantization only consider a fraction of features in the database, the gain of match expansion in this case depends on the desired recall of the index structure.

Note that, while such a match expansion can find additional hypotheses for candidate images, i.e. increase *MAP*, it cannot retrieve any new candidates, i.e. increase recall which has to be ensured during feature matching. This stage therefore aims at compensating for the loss in MAP due to querying less features. Expansion exploits the keypoint information of the seed matches that provide scale, rotation, and possibly affine information. These properties can be used to identify spatially close keypoints [5, 7, 34, 35, 41]. Given that a match hypothesis is correct, not only the corresponding feature pair should match, but also its spatial neighborhood. The similarity of a match's neighborhood is evaluated using the procedure shown in Figure 2. The figure shows an initial seed match, i.e. a kNN of a query feature and keypoints surrounding the seed match. The size of each keypoint is represented by the icon diameter, and the gradient direction is represented by a line anchored in the icon's center. The top row of this figure visualizes the features of the query image I_q, while the bottom row visualizes the image features of a tentative match image I_{db}. Starting point is an initial correspondence pair (p_q^i, p_{db}^j) established by kNN-search in feature space, see Figure 2 a). In a first step, features in a given spatial range are retrieved in the image I_q for p_q^i and in Image I_{db} for p_{db}^j, see Figure 2 b); the spatial range is visualized by a dotted circle. Given the constant δ_{xy}, the spatial range is given by $s_q^i \delta_{xy}$ for the query feature and $s_{db}^j \delta_{xy}$ for the matching database feature (with s_x^y the size of keypoint y in image x), achieving scale invariance. Spatially close keypoints with a significantly different scale (determined by the scale ratio threshold δ_s) than their reference feature are discarded (see the small features in the figure) similar to [5], resulting in two sets of features P_q and P_{db}. These remaining features are rotation-normalized using the reference keypoint's gradient orientation information r_q^i and r_{db}^i, rotating the set of keypoints and their corresponding gradient orientations, see Figure 2 c). Then the two lists of keypoints are traversed in parallel. If the rotation-normalized angle α to the reference feature (i.e. the angular position of the feature to the reference), the rotation-

normalized gradient angle r, and the feature-space distance of two features d_v are within a predefined threshold (δ_α, δ_r, and δ_{d_v} respectively) and the ratio of their scale-normalized spatial distance is within given bounds $\delta_{d_{xy}}$, the corresponding features are accepted as a matching pair (see Figure 2 d)). The remaining features are discarded. Note that, while the complexity of this step is $|P_q| * |P_{db}|$ in the worst case, it can be reduced by an efficient sweep-line implementation that sorts features by their angle α and traverses both lists in parallel. The matching procedure can be called recursively on the expanded features to compensate for non-rigid transforms (in our case with a maximum recursion depth of 2); however, as we will see in the experimental evaluation, the gain of this recursive expansion is relatively low. By choosing the Mahalanobis distance using the affinity matrices of the seed pair (A^i_q and A^i_{db} of affine keypoint detectors) instead of Euclidean distances for finding spatially neighboring keypoints, the process can be extended to affine-invariant features. Result of the expansion phase is an extended list of match hypotheses.

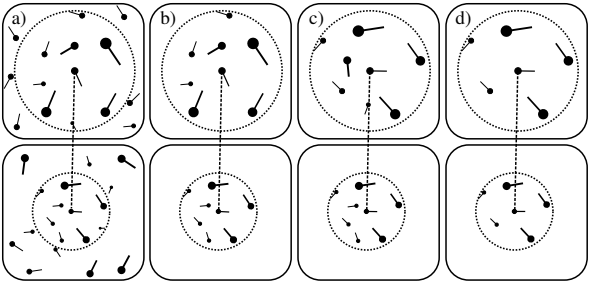

Figure 2: Generation of additional match hypotheses.

Note that the match expansion phase trades a higher memory consumption for effectiveness: In contrast to approaches not relying on geometric information, keypoint information such as the keypoints' position and rotation has to be stored in the database. In contrast to other approaches only relying on geometric verification there is also some additional cost for storing the feature vectors in the database. To mitigate the problem of storing feature vectors, we will consider effective compression techniques after the description of the pipeline.

5) Scoring. Scoring is again tightly interleaved with match generation. Based on the expanded list of matches, a score is computed for every database image. We adapt the technique from [18], weighting scores based on their distance to the query feature and the number of features in the image. For each matched feature from image I_x its score is increased by $(\sqrt{d_{kNN} - d_{ref}})/(\sqrt{|I_q|}\sqrt{|I_x|})$ with d_{kNN} the kNN distance of the seed feature, and d_{ref} the distance between the seed feature and its tentative match in the candidate image, i.e. features generated during match expansion are assigned the same score as their seed match. This score is similar to the scores from [18], however we have added additional square root weighting which further increased effectiveness of these scores. For scoring we implemented a simple burst removal [16] scheme after match expansion that allows only for one correspondence per feature in the query image.

Database. In the most basic case, the image database used for query processing can be seen as a list of tuples $(p^0_0, \ldots, p^{|I_0|}_0, \ldots, p^0_i, \ldots, p^{|I_i|}_i, \ldots)$ where p^j_i denotes a tuple containing feature i from image j and its keypoint information. Features in the list are ordered by their corresponding image to allow efficient match expansion. The memory footprint of the image database can be reduced by compressing the feature vectors used during match expansion: For compressing real-valued feature vectors, we consider Product Quantization as in the matching step (see [17] and Step 3)). Therefore, Product Quantization is used two times in the pipelines in different ways, once for matching and once for match expansion: In contrast to the Locally Optimized Product Quantization based indexing we are using, we do not product quantize residual vectors during the expansion step, but rather the vectors themselves, as otherwise

Extractor	Train	δ_{xy}	δ_s	δ_{d_v}	δ_α	δ_r	$\delta_{d_{xy}}$
SIFT	P6k	6	0.8	26.2	24.3	–	0.49
SIFT	O5k	6	0.8	26.9	18.9	–	0.56
BinBoost	P6k	4	0.8	73	21.1	26.0	0.46

Table 1: Parameters for Match Expansion

vectors belonging to different cells in the outer quantizer could not be compared efficiently. For compression, we split each feature vector in a set of $m = 8$ subquantizers and for each of these subquantizers build a codebook of $s = 256$ centroids. The distance between feature vectors can then easily be approximated as the sum of squared distances between the closest subquantizer centroids followed by a square root operation. As distances between cluster centroids can be stored in a lookup table of size $m * s * s$, distance computations reduce to m table lookups and a single square root operation. Memory consumption could be further minimized by quantizing keypoint information similar to [27].

5 Experiments

5.1 Experimental Setup

We evaluated the modified recognition pipeline on four datasets, *Oxford5k* (O5k) [28], *Oxford105k* (O105k) [28], *Paris6k* (P6k) [29], and *INRIA Holidays* (Hol) [15]. Images of the Holidays dataset were scaled down to a maximum side length of 1024 before feature extraction. We used two different feature extraction techniques: a rotation-variant version of SIFT using affine invariant keypoints[1] from the authors of [27] and, as an instance of state-of-the-art binary descriptors, the BinBoost descriptor which is also publicly available [39]. Concerning *Hessian-affine SIFT*, scale was separated from the affinity matrices according to [27], however for expanding matches we used the square root of this scale which roughly corresponds to the radius of the image patch used for SIFT extraction. These vectors were square-root weighted similar to RootSift [1], however without applying L_1 normalization before taking the element-wise square root. The weighted features were then indexed using LOPQ in combination with a multi-index [21]. We use a vocabulary of size $V = 2 * 1024$ for the inverted lists, and 8 subquantizers for vector quantization, each subquantizer with a vocabulary size of 256 clusters. To compress the feature vectors for the expansion phase, we again used 8 subquantizers consisting of 256 clusters, reducing storage overhead of feature vectors to 6.25% of their uncompressed memory footprint. Codebooks for the Oxford and Holidays dataset were trained on Paris6k, and for Paris6k codebooks where trained on Oxford5k. During query processing, we applied burst removal [16]. To index *BinBoost* descriptors we used Multi-Index Hashing [26]. The code was written in C++ using OpenCV. Runtime experiments were conducted on an off-the-shelf Linux Machine with i7-3770@3.40GHz CPU and 32GB of main memory without parallelization.

Parameters. Match expansion parameters were set as follows. First, range multiplier δ_{xy}, maximum scale change δ_s, k, and n were set by hand with computational efficiency in mind, as a lower number of features considered during expansion reduces the cost of this step. Given these manually set parameters, the remaining parameters of the expansion phase, i.e. feature distance threshold δ_{d_v}, angular threshold δ_α, gradient angle threshold δ_r and spatial distance ratio $\delta_{d_{xy}}$ were set to the outcome of a *Nelder-Mead Downhill-Simplex optimization* maximizing MAP; initialization was performed with reasonable seed values. Optimization

[1] https://github.com/perdoch/hesaff/

(a) SIFT, Oxford5k, k=100

↓ Appr. → n	50	100	500	1000
RND	.616	.698	.810	.827
RESP	.557	.640	.787	.822
ANMS	.676	.727	.825	.836
RND+ME	.679	.749	.829	.838
ANMS+ME	.741	.780	.843	.844
RND+MER	.686	.752	.826	.832
ANMS+MER	.752	.786	.837	.838

(b) BinBoost, Oxford5k, k=100

↓ Appr. → n	50	100	500	1000
RND	.390	.462	.586	.616
RESP	.389	.461	.600	.625
ANMS	.461	.508	.614	.620
RND+ME	.469	.529	.625	.638
ANMS+ME	.542	.588	.648	.644
RND+MER	.481	.539	.626	.634
ANMS+MER	.551	.591	.648	.640

Table 2: Oxford5k: SIFT and BinBoost

(a) SIFT, Holidays, k=10

↓ Appr. → n	50	100	500	1000
RND	.600	.662	.765	.792
RESP	.571	.630	.735	.770
ANMS	.642	.696	.779	.803
RND+ME	.646	.702	.764	.770
ANMS+ME	.699	.734	.780	.781

(b) SIFT, Paris6k, k=100

↓ Appr. → n	50	100	500	1000
RND	.566	.652	.770	.786
RESP	.519	.594	.743	.775
ANMS	.578	.668	.783	.794
RND+ME	.629	.699	.781	.789
ANMS+ME	.648	.723	.793	.796

Table 3: Holidays and Paris6k on SIFT

was done on the Paris6k dataset (with LOPQ and quantization codebooks trained on Paris6k as well) for the Oxford5k, Oxford105k and Holidays datasets. For Paris6k, we optimized on Oxford5k. Parameters were selected for each of the descriptor types (SIFT and BinBoost) using ANMS ranking at $k = 100$, $n = 10$, recursively descending into every expanded match. The resulting parameters were reused for the remaining ranking approaches, different k, n and the non-recursive approach. An overview over the selected parameters is shown in Table 1.

We varied each of the optimized parameters by $\pm 10\%$ separately on Oxford 5k (ANMS ranking with match expansion) to get insights into their effect on MAP. The maximum deviation resulted from decreasing the feature distance threshold, which lead to a decrease in MAP of -0.012, indicating that while there is an impact of the optimized parameters on the performance of match expansion, there is still a range of relatively "good" parameters.

5.2 Experiments

We evaluated the algorithm's performance by varying k and n as these parameters affect the number of initial seed points that are expanded later. Our baselines without match expansion use the C_a+*SRN* scoring from [18]. Recall that n upper bounds, for a query image, the number of kNN queries.

Keypoint Ranking. Our first experiment (see Table 2a and Table 2b) evaluates the performance difference in MAP of different keypoints ranking techniques when querying a low number of features (i.e. 50, 100, 500, 1000 keypoints) to select the best one for match expansion. We evaluated a random baseline (RND), ranking by keypoint responses (RESP), and Adaptive Non-Maximal Suppression [3] (ANMS). The MAP of the response-based ranking is worse or similar to the random baseline. In contrast ANMS increases MAP for all approaches. Note that the gain resulting from using ANMS is rather astonishing for Oxford5k (Table 2a) and Holidays (Table 3a). For Paris6k (Table 3b), the gain is lower. For BinBoost (Table 2b), ANMS can increase performance on Oxford5k as well. The average number of retrieved images for ANMS at 50 keypoints is approximately 2450 and increases to about 4800 images at 1000 keypoints.

(a) SIFT, Oxford 105k, k=100

↓ Appr. → n	50	100	500	1000
ANMS	.489	.554	.710	.748
ANMS+ME	.584	.630	.753	.775

(b) BinBoost, Oxford 105k, k=100

↓ Appr. → n	50	100	500	1000
ANMS	.369	.412	.527	.558
ANMS+ME	.445	.477	.576	.590

Table 4: Oxford105k: SIFT and BinBoost

Match expansion. Our second experiment aims at evaluating the gain in MAP that can be achieved for a low number of kNN queries when additional hypotheses are generated by match expansion (ME) and the same approach in its recursive version (MER). Affine-invariant SIFT features (ANMS+ME, n=50) achieve about 90% of the random baseline (RND,n=1000) on Oxford5k, where the baseline (RND, n=50) only achieves 75%. At the same time the results at 1000 keypoints are similar for all approaches, showing that match expansion does not considerably affect MAP if a high number of keypoints is queried. This substantiates our statement made in the introduction: if a small number of features is queried, techniques that do not achieve high performance gain for a high number of features can achieve considerable gain in performance. Results are similar for Holidays (88% for ANMS+ME@$n = 50$ vs. 76% for the random baseline) while for Paris6k the gain of match expansion is lower (82% vs 72% for the random baseline). Further note that MAP for ANMS+ME decreases slower with decreasing n than without expansion (-.003 (ANMS+ME) vs. -.011 (ANMS) for $n : 1000 \rightarrow 500$ on Paris6k). Additional results for Oxford105k are shown in Table 4a. For BinBoost (Table 2b and Table 4b), match expansion improves results as well. While there is some gain for recursively descending (MER) into matches, this additional step does not significantly improve the performance with both SIFT and BinBoost, while being computationally much more expensive. ANMS+ME on Oxford using SIFT accepted about 16500 matches per query image ($n = 100$, $k = 100$), in contrast to the approximately 8800 tentative correspondences (less than $n * k$ due to burst removal) that have been generated using kNN matching alone (ANMS).

Value of k. As explained in the main section of this paper, not only the number of queried keypoints can be used to increase the number of seed hypotheses and therefore the matching quality, but also k. What happens to the optimal value of k when we decrease the number of keypoints? As shown in Figure 3, if a large number of keypoints is queried ($n = 1000$), then for all of the evaluated approaches a value of $k = 100$ performed better than $k = 1000$. So in general, match expansion does not affect the optimal value of k in this case. However, if only very few keypoints are used for query processing (e.g. $n = 10$), a large k performed better with match expansion. Without the additional ME step, performance decreased for large k (however at a larger k than at a higher number of keypoints queried), as the additional false correspondences could not be out-weighted by the higher number of correct matches. This leads us to the following results: The best way to increase query performance, which is well known, is to increase the number of keypoints queried. In order to increase query efficiency however, it is possible to decrease the number of keypoints queried. In this case, some of the performance loss resulting from a lower number of keypoints can be

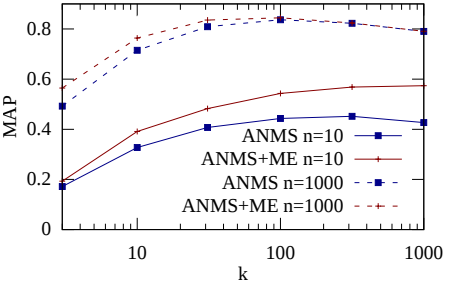

Figure 3: MAP for varying k (SIFT, O5k)

compensated by a large k in combination with match expansion (and, at a lower degree, even without ME).

Runtime. The cost of the evaluated *keypoint ranking* approaches is negligible for the random and response based ones, as these just have to sort the query features, and about 7ms (SIFT) and 5ms (BinBoost) for the ANMS ranker (Oxford5k). For SIFT, scoring times (including ME) were about 6ms for processing all k results of a single kNN query ($k = 100$, $n = 100$), and therefore slightly lower than the runtimes of running a single kNN query which took about 7ms, at the possible gain of adding additional matches. The feature quantization needed for match expansion took about 45ms for all features in a query image. For BinBoost features (including ME), the match expansion and scoring took less than 4ms for processing a single kNN result. The overall runtime for Hessian-Affine SIFT at 100 keypoints (ANMS+ME) was about 1.34s ($k = 100$,$n = 100$), while for binary features it was significantly higher (15s), as for this we used an exact, though state-of-the-art, indexing technique. Setting runtimes in relation to MAP, it is possible to beat an RND ranker considering 100 keypoints with ANMS+ME considering 50 keypoints at a slightly lower runtime 0.69s vs 0.75s and a higher MAP (see Table 2a, .698 vs. .741, O5k). For the holidays dataset runtimes of the random approach (RND) were about 0.9s ($k = 10$, $n = 100$) and for ANMS+ME it was only approximately 0.6s ($k = 10$, $n = 50$) at a higher MAP. The most time-consuming operation during match expansion is the search of spatially close features; we think that the runtimes of match expansion can be reduced significantly by optimizing this matching step. For Oxford105k the runtime for match expansion was similar to Oxford5k: A single kNN query took slightly less than 8ms and expansion took about 7ms. Runtimes have been measured using only a single core, however algorithms can be easily parallelized.

Comparison to the State of the Art. While the primary goal of this research is not to increase the effectiveness of object recognition but rather to reduce the number of features queried, let us still compare our results to the state of the art in order to get insights into its performance. We compare to [30], as the authors were using the same SIFT features and a recognition pipeline involving Product Quantization. On Oxford5k [30] achieved MAP of 0.78 using Product Quantization with 8 subquantizers, i.e. a setting close to our scenario. This corresponds to the performance we could achieve when querying 100 keypoints. However, using the pipeline from our paper requires a larger memory footprint; if we consider techniques with a memory footprint closer to ours, [30] was able to achieve 0.83 points in map by approximating features more accurately using 32 bytes per feature, lower than ANMS+ME@n=500.

6 Conclusion

In this paper we evaluated an alternative pipeline for decreasing the runtimes of object recognition when kNN queries are used for the generation of tentative correspondences instead of Bags of Visual Words. While the reduction of query features can have negative effects on query performance, especially if the unmodified standard recognition pipeline is used, some simple modifications in the pipeline aiming at feature ranking and match expansion can already produce good results at only a fraction of kNN queries. Further improvements in the match expansion stage should aim at increasing its efficiency and effectiveness. Due to the simple structure of the pipeline used in this paper, such improvements can be easily integrated.

References

[1] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proc. CVPR*, pages 2911–2918. IEEE, 2012.

[2] Artem Babenko and Victor Lempitsky. The inverted multi-index. In *Proc. CVPR*, pages 3069–3076, 2012.

[3] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Proc. CVPR*, volume 1, pages 510–517. IEEE, 2005.

[4] 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 *Proc. ICCV*, pages 1–8. IEEE, 2007.

[5] Ondrej Chum, Michal Perdoch, and Jiri Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *Proc. CVPR*, pages 17–24. IEEE, 2009.

[6] Chunhui Cui and King Ngi Ngan. Global propagation of affine invariant features for robust matching. *TIP*, 22(7):2876–2888, 2013.

[7] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Simultaneous object recognition and segmentation by image exploration. In *Proc. ECCV*, pages 40–54. Springer, 2004.

[8] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *Proc. CVPR*, pages 2946–2953, 2013.

[9] Xiaojie Guo and Xiaochun Cao. Good match exploration using triangle constraint. *Pattern Recognition Letters*, 33(7):872–881, 2012.

[10] Kiana Hajebi and Hong Zhang. Stopping rules for bag-of-words image search and its application in appearance-based localization. *arXiv preprint arXiv:1312.7414*, 2013.

[11] Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Predicting matchability. In *Proc. CVPR*, 2014.

[12] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proc. CVPR*, pages 2938–2945, 2013.

[13] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *Proc. CVPR*, pages 2957–2964, 2012.

[14] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. STOC*, pages 604–613, 1998.

[15] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proc. ECCV*, pages 304–317. Springer, 2008.

[16] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *Proc. CVPR*, pages 1169–1176. IEEE, 2009.

[17] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE PAMI*, 33(1):117–128, 2011.

[18] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Exploiting descriptor distances for precise image search. Technical Report 7656, INRIA, 2011.

[19] Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *Proc. CVPR*, pages 873–880, 2011.

[20] Il-Kyun Jung and Simon Lacroix. A robust interest points matching algorithm. In *Proc. ICCV*, volume 2, pages 538–543. IEEE, 2001.

[21] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proc. CVPR*, 2014.

[22] Seungjin Lee, Kwanho Kim, Joo-Young Kim, Minsu Kim, and Hoi-Jun Yoo. Familiarity based unified visual attention model for fast and robust object recognition. *Pattern Recognition*, 43(3):1116–1128, 2010.

[23] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. Location recognition using prioritized feature matching. In *Proc. ECCV*, pages 791–804. Springer, 2010.

[24] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110, 2004.

[25] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, pages 2161–2168, 2006.

[26] Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In *Proc. CVPR*, pages 3108–3115, 2012.

[27] Michal Perd'och, Ondrej Chum, and Jiri Matas. Efficient representation of local geometry for large scale object retrieval. In *Proc. CVPR*, pages 9–16. IEEE, 2009.

[28] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007.

[29] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. CVPR*, pages 1–8, 2008.

[30] Danfeng Qin, Christian Wengert, and Luc J. Van Gool. Query adaptive similarity for large scale object retrieval. In *Proc. CVPR*, pages 1610–1617, 2013.

[31] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Proc. ICCV*, pages 2564–2571, 2011.

[32] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Proc. ICCV*, pages 667–674. IEEE, 2011.

[33] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. *Proc. ECCV*, pages 752–765, 2012.

[34] Frederik Schaffalitzky and Andrew Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *Proc. ECCV*, pages 414–431. Springer, 2002.

[35] Cordelia Schmid and Roger Mohr. Combining greyvalue invariants with local constraints for object recognition. In *Proc. CVPR*, pages 872–877. IEEE, 1996.

[36] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, pages 1470–1477, 2003.

[37] Giorgos Tolias, Yannis Kalantidis, Yannis Avrithis, and Stefanos Kollias. Towards large-scale geometry indexing by feature selection. *Computer Vision and Image Understanding*, 120:31–45, 2014.

[38] Antonio Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *Proc. CVPR*, 2008.

[39] Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. In *Proc. CVPR*, pages 2874–2881. Ieee, 2013.

[40] Panu Turcot and David G Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *Computer Vision Workshops, 2009*, pages 2109–2116. IEEE, 2009.

[41] Zhong Wu, Qifa Ke, Michael Isard, and Jian Sun. Bundling features for large scale partial-duplicate web image search. In *Proc. CVPR*, pages 25–32. IEEE, 2009.

[42] Wengang Zhou, Yijuan Lu, Houqiang Li, and Qi Tian. Scalar quantization for large scale image search. In *Proc. MM*, pages 169–178, 2012.