SIViP (2016) 10:1527–1534
DOI 10.1007/s11760-016-0966-6

ORIGINAL PAPER

Instance detection by keypoint matching beyond the nearest neighbor

Furkan Eren Uzyıldırım[1] · Mustafa Özuysal[1]

Received: 7 March 2016 / Revised: 20 July 2016 / Accepted: 8 August 2016 / Published online: 20 August 2016

Abstract The binary descriptors are the representation of choice for real-time keypoint matching. However, they suffer from reduced matching rates due to their discrete nature. We propose an approach that can augment their performance by searching in the top K near neighbor matches instead of just the single nearest neighbor one. To pick the correct match out of the K near neighbors, we exploit statistics of descriptor variations collected for each keypoint in an off-line training phase. This is a similar approach to those that learn a patch specific keypoint representation. Unlike these approaches, we only use a keypoint specific score to rank the list of K near neighbors. Since this list can be efficiently computed with approximate nearest neighbor algorithms, our approach scales well to large descriptor sets.

Keywords Computer vision · Keypoint matching · Object detection

1 Introduction

The binary descriptors (such as [1,5,10,11,18,21,22]) are extensively used in real-time applications for object detection and augmented reality. In particular, they are fast to compute, and the distance between two descriptors can be calculated in a few machine instructions. Given a query descriptor, the latter property greatly speeds up the search for the nearest neighbor within a set of reference descriptors.

Unfortunately, the binary descriptors are particularly sensitive to larger changes in viewpoint and scale [20]. This is mainly due to their discrete nature. For large binary descriptor sets, the nearest neighbor may not even be unique. However, it is relatively easy to find the closest K near neighbors by ranking the matches according to their descriptor distances. This list is more likely to contain the correct match compared to the set that includes only the nearest neighbor. In practice, this extra set of match hypotheses is rarely exploited.

In this paper, we propose a two-step approach to keypoint matching with binary descriptors. In the first step, for each query keypoint, we identify the list of the top K near neighbors according to the Hamming distance. In the second step, we rank these near neighbors according to a probabilistic and keypoint specific match quality score. This score exploits precomputed data extracted from the reference image during an off-line training phase. We show that a keypoint specific measure is more effective in ranking the match hypotheses than the Hamming descriptor distance as illustrated in Fig. 2.

To motivate our approach, we illustrate the existence of correct matches beyond the nearest neighbor. Figure 1 shows the number of correct matches between BRIEF descriptors that fall into the first ten near neighbors. Although most of the BRIEF matches obtained by the nearest neighbor matching are wrong, the correct ones are not very far away in the near neighbor list. The matches beyond the nearest neighbor are only reachable if we can supplement the Hamming distance with a secondary and more distinctive match score.

In designing this secondary match score, we make the following observations: Most binary descriptors, even the

This work was supported by the TÜBİTAK project 113E496.

Electronic supplementary material The online version of this article (doi:10.1007/s11760-016-0966-6) contains supplementary material, which is available to authorized users.

✉ Mustafa Özuysal
mustafaozuysal@iyte.edu.tr

Furkan Eren Uzyıldırım
furkanuzyildirim@iyte.edu.tr

[1] Department of Computer Engineering, İzmir Institute of Technology, Izmir, Turkey

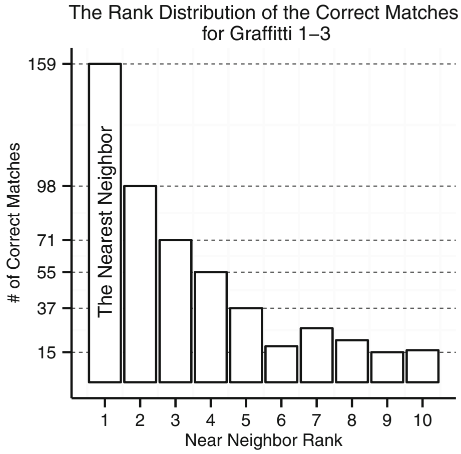


Fig. 1 Near neighbor rank distribution of the correct matches between BRIEF descriptors of the first and the third images of the Graffiti data set. The nearest neighbor obtained by ranking matches according to Hamming distance captures only 159 of the 848 possible correspondences. Meanwhile, the first ten near neighbors include 517 correct matches, a potential improvement by a factor of more than three. In practice, as demonstrated by the results of Fig. 4, 385 of these can be recovered using the approach that we propose, which yields an actual improvement by a factor of more than two

ones that optimize their representation, compute the same features for every keypoint. As shown recently by [3], while computationally appealing, globally optimizing the features for all keypoints is not as effective as picking unique features for each individual keypoint. There are a few existing approaches that learn and use keypoint specific representations [3,8,9,19]. However, these require a separate distance computation to each reference keypoint and can not be directly used with approximate nearest neighbor (ANN) algorithms (such as Locality Sensitive Hashing [2], LSH).

In contrast to these, at the second matching stage, we have only K possible match hypotheses (corresponding to the K near neighbors). We employ a keypoint specific representation to rank these and recover correct matches that did not make it to the top initially. For each reference keypoint k_i in the K near neighbor list, our score computation relies on a precomputed statistical model of the variations of the descriptor bits for keypoint k_i. Since the list of K near neighbors can be computed efficiently with ANN methods, our approach also scales well to larger descriptor sets.

Our main contributions can be summarized as follows:

- We propose a method that is able to find keypoint matches within the list of K near neighbors at negligible additional computational cost at run-time.
- We demonstrate that, despite learning a separate representation for each individual keypoint similar to [3,8,9,19], our approach does not require a brute-force search in a large descriptor set when coupled with the LSH [2] approach to compute the list of K near neighbors.
- We show that our approach is relatively descriptor independent and it extends the matching range of several binary descriptors: BRIEF [5], ORB [21], BRISK [10], FREAK [1], and LATCH [11], which has recently been shown to outperform state-of-the-art binary descriptors.

2 Related work

As discussed in the introduction, a keypoint specific matching approach is shown to perform better than matching with a generic descriptor [3]. One way to achieve this is to compute a descriptor specifically adapted to a given patch. [8] proposed selecting a patch specific set of binary features that are robust to changes in intensity levels as well as small translation and rotations. More recently, [3] proposed to learn a combined set of generic features and a locally optimized binary mask that picks the best features depending on the image patch. While both approaches greatly improve the matching performance, they both require a brute-force search in the reference collection; therefore, they are not suitable for real-time operation on large data sets.

Similarly, the keypoint classification approaches of [9,19] train keypoint specific classifiers and compute a probability distribution over all reference keypoints at run-time. More-

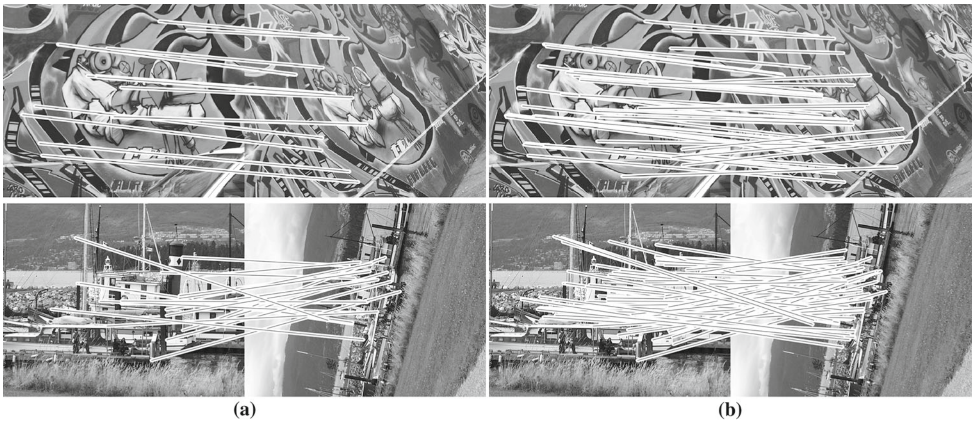


Fig. 2 Matches between the first and the fourth images of the *Graffiti* and *Boat* sequences. **a** Considering only the nearest neighbor with the Hamming distance. **b** Searching within the first ten near neighbors using the approach presented in Sect. 3

over, they require large amounts of memory per keypoint to store the learned probabilities.

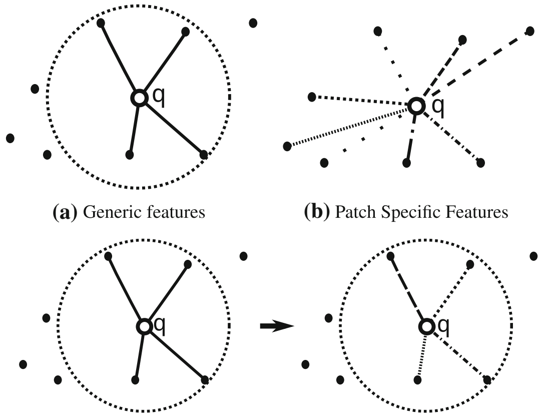
Despite using a probabilistic model similar to that of [9, 19], our approach only makes use of these probabilities for the first ten match candidates. When matching a single keypoint against a database of a thousand reference keypoints, the required number of probability calculations is two orders of magnitude less than those required by Random Ferns.

The combination of a generic initial query followed by candidate specific filtering is quite common in the image retrieval literature [7,12]. For image retrieval, the filtering in the second stage depends on the geometric consistency between the query image and the candidate results. Our approach has a very similar pipeline where the geometry check is replaced by a probabilistic observation probability.

3 A two-step approach to match keypoints

The usual approach for matching keypoints involves locating the nearest neighbor keypoint in descriptor space. Instead of this one-shot approach, we first rank the list of possible matches by their inverse descriptor distance and pick the first few tens of near neighbors to the query in the descriptor space as possible match candidates. We then evaluate each candidate based on detailed statistics of the texture around the specific candidate. Figure 3 gives an overview of the proposed approach.

We learn the statistical model for each keypoint in an off-line training stage by simulating affine deformations and observing the change in the descriptor bits. In the following, we describe the way the descriptor statistics are collected during training and how to score the multiple match hypotheses based on this data.



(a) Generic features

(b) Patch Specific Features

(c) Proposed Approach

Fig. 3 **a** Given a query descriptor q, if the descriptor uses the same features for each patch then it is possible to use an ANN approach to limit distance computation only to a subset of the reference descriptors (*dashed circle*). **b** Some approaches learn and employ a patch specific representation that is more distinctive and robust. **c** We propose a two-step approach that combines the advantages of both by limiting the patch specific scoring to the list of K near neighbors

3.1 Modeling descriptor variations

The binary descriptor bits are not fully invariant to changes in perspective and lighting. Which bits are more prone to variation depends on the texture around each keypoint. If there is a strong gradient near one of the pixels that are part of the computation of the bit's value, then that bit is more likely to flip. Due to the complex nature of these interactions, bit variations are better captured by a probabilistic conditional model such as

$$P(D \mid C = k_i) = P(d_1, d_2, \ldots, d_S \mid C = k_i), \tag{1}$$

where D and C are two random variables corresponding to the descriptor value and the keypoint identity. $C = k_i$ means that the distribution is computed for keypoint i, d_j is a binary random variable representing the jth descriptor bit, and S is the descriptor size in bits.

This representation requires 2^S parameters per keypoint and since S is usually larger than or equal to 64, directly modeling the joint probability of the descriptor bits is not feasible. Following the representation proposed by [19], we split the descriptor into N groups of M bits such that $S = M \times N$ and assume independence between these:

$$P(D \mid C = k_i) = \prod_{j=1}^{N} P(D_j \mid C = k_i), \tag{2}$$

where D_j represents the values of the bits in group j. This representation has $N \times 2^M$ parameters. We use several settings with $M = 4, 6,$ and 8. For 256 dimensional descriptors, these yield between $N = 32$ and 64. Larger M values yield a very large number of parameters and around $M = 12$, we experimentally found that the matching performance degrades. Moreover, a large M means greater memory consumption. In the next section, we provide experimental results for M between 4 and 8. N is determined by fixing M and taking $N = \frac{\text{descriptor size}}{M}$.

The descriptor bits can be assigned to groups in many ways. The simplest possibility is to assign the consecutive descriptor bits to the same group. For descriptors like BRIEF, this is a natural choice as there is no reason to favor one grouping scheme over another. For descriptors like BRISK with more structure, an optimization over possible groupings might be beneficial. In practice, we found that the consecutive grouping works well for all descriptors and we use it in the rest of the experiments.

For each keypoint, we generate training samples using the affine deformation model proposed by [16]. This is not too restrictive since most keypoint detectors assume a locally affine model. In all the experiments, we use the same set of training parameters, scale changes between $\frac{1}{\sqrt{2}}$ and $\sqrt{2}$, in-plane rotation varies between $-30°$ and $+30°$, tilt amount (θ in [16]) varies between 0° and 60°, and tilt angle (ϕ in [16]) varies between 0° and 180°.

We generate roughly 200,000 images, yielding an equivalent number of training descriptors for each keypoint. For each bit group, the probabilities of Eq. 2 are inferred by counting the number of times training descriptors assume a particular value. For example, for bit groups of size 4, we have 16 possible outcomes. We count the number of times each outcome is observed within the training set and normalize by the total number of training samples. To counter the adverse effect of zeros in the estimated probabilities, we start counting from one as suggested by [19], which is equivalent to assuming 2^M pseudo-samples for each keypoint taking on every possible descriptor value within a bit group. This is usually referred to as Laplace smoothing [13] and corresponds to assuming a Dirichlet prior when a Bayesian estimate is made for the probabilities of Eq. 2 [4].

3.2 Keypoint specific scoring of match hypotheses

For each keypoint, we first obtain the list of K near neighbors based on the Hamming distance between the descriptors. This list is then sorted according to a score specific to the particular candidate keypoint each near neighbor represents.

We have experimented with various functions to score each hypothesis by combining the descriptor statistics and the original Hamming distance in several different ways. The best results have been obtained when the score is the negative Hamming distance between the query and reference descriptor plus the logarithm of the probability of observing the query descriptor according to Eq. 2:

$$\mathrm{score}(Q, k_i) = -\left|Q - D^i\right|_H + \log \prod_{j=1}^{N} P\left(Q_j \mid C = k_i\right), \tag{3}$$

where Q is the query descriptor, D^i is the reference descriptor i, and $|X - Y|_H$ is the Hamming distance between X and Y.

Since the second term involves k_i, its direct evaluation would require a number of operations linear in the reference data set size. Our main insight is that computing these for the first K near neighbors is sufficiently effective in recovering many more matches than using only the nearest neighbor.

To demonstrate this and the effect of K on the recognition performance, we perform a preliminary experiment on the *Graffiti* and *Wall* sequences [14,15]. We detect approximately 1000 keypoints on the reference image and transform their coordinates to the rest of the images using the ground truth homography. We compute BRIEF descriptors around the reference and the test keypoints. For each test descriptor, we compute the K near neighbors using Hamming distance and then pick the best match according to Eq. 3. We measure and report the recognition rate as the percentage of test keypoints that are matched to the correct reference keypoints for various K values with bit group size M set to 8. The results are given in Fig. 4. Regardless of descriptor type, enlarging the near neighbor list yields improved recognition rates. The improvement is less pronounced after $K = 10$.

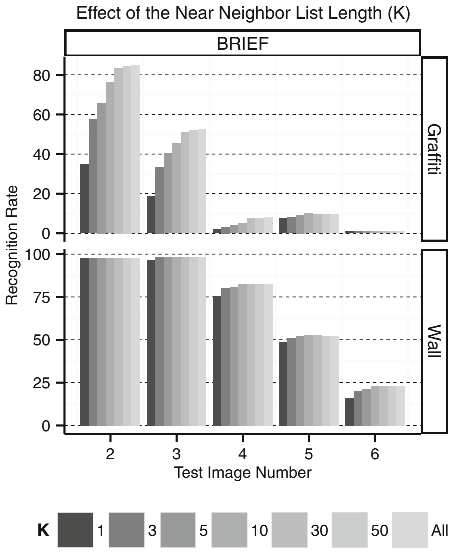


Fig. 4 Recognition rates for BRIEF increase as we consider more near neighbors (NN). This is especially true for *Graffiti*, where the baseline performance ($K = 1$) is low and the test images exhibit both scale and rotation changes. The difference between considering the first ten NN or all NNs is relatively low. At $K = 10$, the number of correct matches for Graffiti-3 increases to 385. This is still less than the maximum potential number 517 (See Fig. 1), but substantially better than 159, the nearest neighbor baseline

4 Experiments

To test the ability of our approach in recovering the correct matches from the near neighbor list, we have performed three sets of experiments. The first one follows the experimental setup of Fig. 4, and we report the recognition rate over ground truth correspondences. The second one measures the inlier ratio as a function of the number of keypoints matched in the test images. The final experiment measures the recognition rate when there are a larger number of reference images and an approximate nearest neighbor algorithm is used to compute the near neighbor list.

4.1 Recognition rate over ground truth correspondences

We repeat the experiments of Fig. 4 for four data sets and five descriptor types using $K = 10$ and three values for the

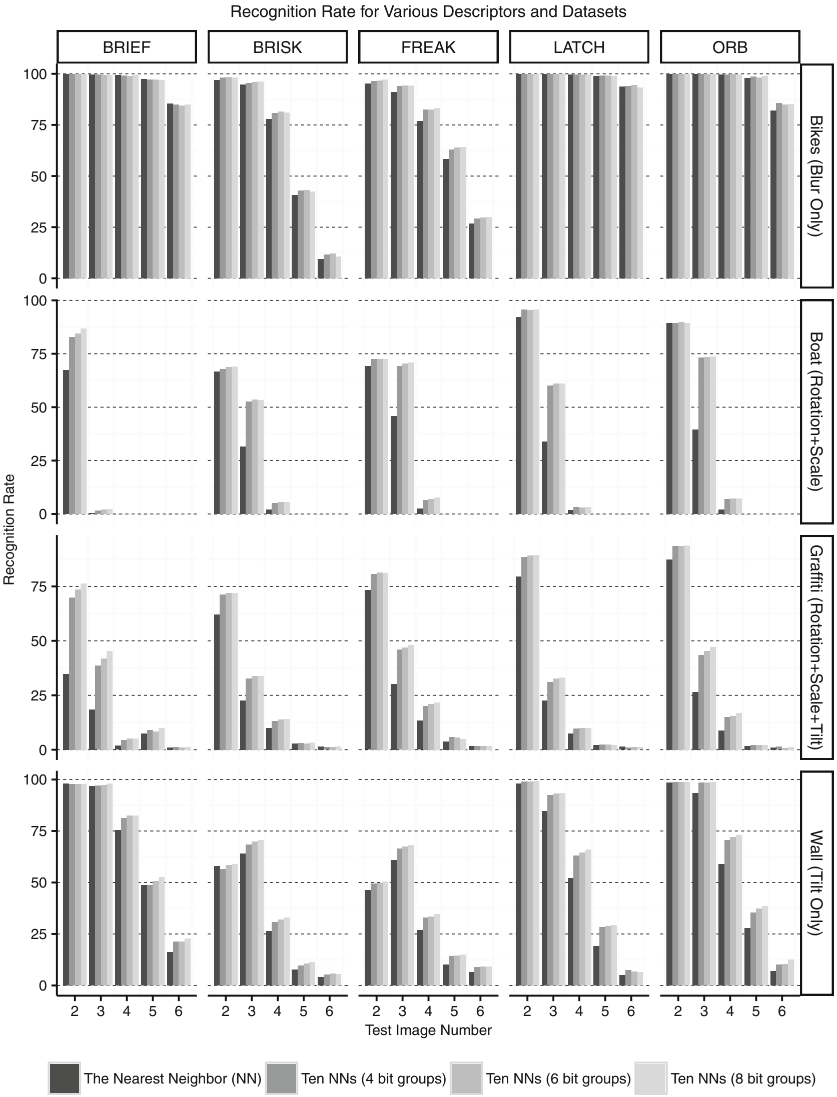


Fig. 5 Recognition rate improves when we exploit the information in the first ten near neighbors as opposed to using just the nearest neighbor. Note that the improvement is most significant when the recognition rate is low (3–6) and the number of matches may not be enough for correct registration of the test image. The performance improves as M is increased from 4 to 8, however increasing M beyond 10 degrades performance due to the exponentially increasing number of parameters in the probabilistic model that requires even more training data. Hence we limit our experiments to $M = 4$ and $M = 8$

bit group size ($M \in \{4, 6, 8\}$). The results[1] are shown in Fig. 5. Our approach increases the recognition rate particularly when the data set includes larger changes in perspective. For the *Bikes* sequence with only changes in blur level, the improvement is less pronounced. This is expected since the observation probabilities represent behavior under perspective changes. We have tried including blur in the training data, but this did not yield noticeable improvement, possibly because blur causes loss of discriminative power.

BRIEF lacks orientation estimation, as a result after the test image 2 of the *Boat* sequence even the top ten near neighbor list does not include enough correct matches and our approach does not improve recognition rate. For descriptors with orientation estimation such as ORB, the results are improved by a large amount for the test image 3. The most significant improvement is achieved for the *Graffiti* data set which contains scale and perspective changes.

4.2 Improving the inlier ratio characteristics

The previous experiments measure the classification performance of the keypoint matching over ground truth correspondences. As a more realistic test, we detect keypoints in the test images and we match each one to the reference keypoints to yield a list of potential correspondences. We rank these by their negative descriptor distance and measure the ratio of the correct matches to the total number of matches. This ratio is equal to the *inlier ratio* during the iterations of a robust estimator such as PROSAC [6], and it is a measure of the precision of the keypoint matching approach. The inlier ratio values obtained as such directly influences the required

[1] Note that for each descriptor type, we use different keypoint detector and descriptor settings (either the OpenCV defaults or the setup used by the authors). So, the figures in this paper should not be used for performance comparison between different descriptors.

Table 1 Inlier ratio values when matching the third test image to the reference image in each image

Descriptor	BRIEF			BRISK			FREAK			LATCH			ORB		
# of matches	100	250	500	100	250	500	100	250	500	100	250	500	100	250	500
Bikes															
NN	**0.97**	**0.97**	**0.88**	0.78	0.61	0.46	**0.86**	0.72	0.53	**0.99**	0.95	0.70	**0.99**	0.97	–
RNN-4 bits	0.93	0.90	0.85	**0.84**	0.73	0.51	0.85	0.79	0.58	0.96	0.95	0.70	0.98	0.97	–
RNN-8 bits	0.94	0.91	0.87	0.81	0.70	0.51	**0.86**	0.79	0.59	0.98	**0.96**	0.70	0.98	0.97	–
5NN-4 bits	0.94	0.91	0.86	**0.84**	0.74	0.59	0.84	0.80	0.62	0.96	**0.96**	**0.72**	**0.99**	0.97	–
5NN-8 bits	0.95	0.92	**0.88**	0.80	0.76	**0.60**	0.85	**0.81**	**0.63**	0.98	**0.96**	**0.72**	**0.99**	**0.98**	–
10NN-4 bits	0.94	0.91	0.86	**0.84**	0.74	**0.60**	0.84	0.80	0.62	0.96	**0.96**	**0.72**	**0.99**	0.97	–
10NN-8 bits	0.95	0.92	**0.88**	0.80	**0.77**	**0.60**	0.85	**0.81**	**0.63**	0.98	**0.96**	**0.72**	**0.99**	**0.98**	–
20NN-4 bits	0.94	0.91	0.86	**0.84**	0.74	**0.60**	0.84	0.80	**0.63**	0.96	**0.96**	**0.72**	**0.99**	0.97	–
20NN-8 bits	0.95	0.92	**0.88**	0.80	**0.77**	**0.60**	0.85	**0.81**	**0.63**	0.98	**0.96**	**0.72**	**0.99**	**0.98**	–
Boat															
NN	0.00	0.00	0.00	0.65	0.44	0.29	0.60	0.45	0.37	0.40	0.22	0.13	0.13	0.14	0.11
RNN-4 bits	0.00	0.00	0.00	0.94	0.63	0.35	0.87	0.75	0.48	0.54	0.27	0.14	0.51	0.28	0.15
RNN-8 bits	0.00	0.00	0.00	0.96	0.65	0.35	**0.90**	0.77	0.49	0.61	0.27	0.14	0.55	0.28	0.15
5NN-4 bits	0.00	0.00	0.01	0.96	0.78	0.52	0.85	0.80	0.65	0.66	0.44	0.23	0.64	0.46	0.29
5NN-8 bits	0.00	0.00	0.01	**0.98**	0.82	0.53	0.89	**0.83**	0.67	0.71	0.44	0.24	0.72	0.50	0.29
10NN-4 bits	0.00	0.00	0.02	0.96	0.80	0.56	0.85	0.81	0.67	0.68	0.50	0.27	0.67	0.50	0.34
10NN-8 bits	0.01	0.01	0.03	**0.98**	**0.86**	0.59	0.89	**0.83**	0.70	0.74	0.52	0.29	**0.76**	0.54	0.36
20NN-4 bits	0.00	0.00	0.01	0.96	0.80	0.59	0.85	0.81	0.67	0.69	0.51	0.29	0.67	0.50	0.37
20NN-8 bits	0.01	0.02	0.03	**0.98**	**0.86**	**0.63**	0.89	**0.83**	**0.71**	**0.76**	**0.56**	**0.32**	0.75	**0.55**	**0.39**
Graffiti															
NN	0.28	0.21	0.16	0.61	0.36	0.23	0.53	0.40	0.30	0.43	0.27	0.17	0.37	0.26	0.18
RNN-4 bits	0.49	0.32	0.20	0.70	0.44	0.26	0.70	0.54	0.34	0.54	0.36	0.19	0.55	0.38	0.22
RNN-8 bits	0.54	0.34	0.20	**0.72**	0.44	0.26	0.72	0.54	0.35	0.53	0.37	0.19	**0.59**	0.39	0.22
5NN-4 bits	0.54	0.44	0.34	0.70	0.50	0.33	0.72	0.57	0.43	0.51	0.37	0.22	0.52	0.40	0.27
5NN-8 bits	0.67	0.52	0.37	**0.72**	0.53	0.35	0.73	0.61	0.43	0.54	0.39	0.24	0.54	0.42	0.29
10NN-4 bits	0.54	0.44	0.35	0.70	0.50	0.34	0.73	0.58	0.44	0.51	0.40	0.23	0.52	0.40	0.27
10NN-8 bits	0.68	0.54	0.41	**0.72**	0.53	0.36	**0.74**	**0.62**	0.46	**0.55**	**0.41**	0.26	0.53	0.42	0.30
20NN-4 bits	0.54	0.46	0.38	0.70	0.50	0.35	0.73	0.59	0.45	0.51	0.39	0.24	0.52	0.40	0.28
20NN-8 bits	**0.69**	**0.56**	**0.45**	**0.72**	**0.54**	**0.38**	**0.74**	**0.62**	**0.47**	**0.55**	**0.41**	**0.27**	0.53	**0.43**	**0.31**
Wall															
NN	0.99	**0.99**	0.88	**1.00**	0.96	0.67	0.97	0.86	0.56	0.90	0.78	0.47	**1.00**	0.95	0.62
RNN-4 bits	0.99	**0.99**	0.90	**1.00**	0.97	0.73	**0.98**	0.91	0.61	0.89	0.84	0.52	**1.00**	**0.96**	0.64
RNN-8 bits	0.99	**0.99**	**0.92**	**1.00**	**0.98**	0.74	0.97	0.91	0.61	0.91	0.85	0.52	**1.00**	**0.96**	0.64
5NN-4 bits	**1.00**	**0.99**	0.89	**1.00**	0.97	0.75	**0.98**	0.90	0.64	0.90	0.84	0.56	**1.00**	**0.96**	0.65
5NN-8 bits	**1.00**	**0.99**	**0.92**	**1.00**	**0.98**	0.78	0.97	**0.92**	0.66	**0.92**	**0.87**	0.56	**1.00**	**0.96**	**0.67**
10NN-4 bits	**1.00**	**0.99**	0.89	**1.00**	0.97	0.75	**0.98**	0.90	0.65	0.90	0.84	0.55	**1.00**	**0.96**	0.65
10NN-8 bits	**1.00**	**0.99**	**0.92**	**1.00**	**0.98**	**0.79**	0.97	**0.92**	**0.67**	**0.92**	**0.87**	**0.57**	**1.00**	**0.96**	**0.67**
20NN-4 bits	**1.00**	**0.99**	0.89	**1.00**	0.97	0.76	**0.98**	0.90	0.65	0.90	0.84	0.55	**1.00**	**0.96**	0.65
20NN-8 bits	**1.00**	**0.99**	**0.92**	**1.00**	**0.98**	**0.79**	0.97	**0.92**	**0.67**	**0.92**	**0.87**	0.56	**1.00**	**0.96**	**0.67**

Bold values indicate the maximum values obtained for each test image (Bikes, Boat, Graffiti, or Wall) within the corresponding column

Results are obtained by ranking the nearest neighbor matches only by the descriptor distance (NN), by the keypoint specific score of Eq. 3 (Reweighted NN → RNN), or ranking the K near neighbor list matches by Eq. 3 (KNN, with $K \in \{5, 10, 20\}$). For each dataset and descriptor combination, inlier ratios at 100, 250, and 500 keypoint matches are listed. In general, keypoint specific ranking improves the inlier ratio for the best few hundred correspondences and looking inside the ten near neighbor list yields improved inlier ratio at 500 keypoint matches

Table 2 With eight reference images, we measure the inlier ratio values when matching the third test image of each image sequence and LSH is used to compute the near neighbor list

Descriptor	BRIEF			BRISK			FREAK			LATCH			ORB		
# of matches	100	250	500	100	250	500	100	250	500	100	250	500	100	250	500
Bikes															
NN	**0.97**	**0.96**	**0.87**	0.53	0.37	0.27	0.81	0.59	0.40	**0.99**	0.95	0.69	**0.99**	0.94	–
RNN-4 bits	0.93	0.90	0.84	0.66	0.49	0.33	0.84	0.68	0.46	0.96	0.95	0.69	0.98	0.96	–
RNN-8 bits	0.94	0.91	0.86	0.64	0.47	0.33	**0.86**	0.70	0.47	0.98	**0.96**	0.69	0.98	0.96	–
10NN-4 bits	0.94	0.91	0.83	**0.70**	0.52	0.40	0.83	0.71	0.51	0.96	**0.96**	**0.71**	**0.99**	**0.97**	–
10NN-8 bits	0.95	0.92	0.86	**0.70**	**0.56**	**0.41**	0.84	**0.72**	**0.53**	0.98	**0.96**	**0.71**	**0.99**	**0.97**	–
Boat															
NN	0.01	0.01	0.00	0.40	0.25	0.15	0.35	0.28	0.20	0.21	0.13	0.08	0.04	0.04	0.04
RNN-4 bits	0.00	0.01	0.00	0.72	0.34	0.17	0.74	0.48	0.27	0.41	0.18	0.09	0.25	0.12	0.06
RNN-8 bits	0.01	0.00	0.00	0.71	0.34	0.17	0.76	0.49	0.27	0.45	0.18	0.09	0.26	0.12	0.06
10NN-4 bits	0.00	0.00	0.00	0.88	0.59	0.35	0.76	0.68	0.50	0.59	0.31	**0.17**	0.46	0.28	**0.17**
10NN-8 bits	0.00	0.00	0.00	**0.91**	**0.64**	**0.36**	**0.82**	**0.74**	**0.51**	**0.64**	**0.32**	**0.17**	**0.49**	**0.30**	**0.17**
Graffiti															
NN	0.14	0.10	0.07	0.59	0.34	0.21	0.47	0.31	0.22	0.39	0.21	0.13	0.32	0.20	0.15
RNN-4 bits	0.27	0.16	0.10	0.65	0.39	0.23	0.64	0.42	0.27	0.46	0.26	0.14	0.46	0.31	0.17
RNN-8 bits	0.29	0.18	0.10	0.69	0.41	0.23	0.66	0.43	0.27	0.47	0.26	0.14	0.48	0.31	0.17
10NN-4 bits	0.35	0.24	0.19	0.66	0.47	0.31	0.67	0.49	0.34	0.42	0.26	**0.17**	0.42	0.30	0.20
10NN-8 bits	**0.46**	**0.33**	**0.23**	**0.70**	**0.48**	**0.32**	**0.68**	**0.52**	**0.37**	**0.49**	**0.29**	**0.17**	**0.46**	**0.33**	**0.21**
Wall															
NN	0.99	**0.98**	0.80	**1.00**	0.88	0.56	0.92	0.68	0.45	0.90	0.69	0.41	0.98	0.82	0.52
RNN-4 bits	0.99	0.96	0.86	**1.00**	0.97	0.60	**0.96**	0.80	0.52	0.89	0.81	0.45	0.98	**0.94**	0.56
RNN-8 bits	0.99	**0.98**	**0.88**	**1.00**	**0.98**	0.61	0.95	0.83	0.52	0.91	0.82	0.45	**0.99**	**0.94**	0.56
10NN-4 bits	0.99	0.96	0.86	**1.00**	0.95	0.66	**0.96**	0.81	0.59	0.90	0.83	0.50	0.98	0.92	**0.59**
10NN-8 bits	**1.00**	**0.98**	**0.88**	**1.00**	0.97	**0.68**	0.95	**0.84**	**0.60**	**0.92**	**0.85**	**0.52**	**0.99**	**0.94**	**0.59**

Bold values indicate the maximum values obtained for each test image (Bikes, Boat, Graffiti, or Wall) within the corresponding column

Similar to the results of Table 1, ranking the correspondences by Eq. 3 improves the overall values. Searching in the first ten near neighbors further increases the number of inliers that can be obtained

minimum number of PROSAC iterations to correctly calculate the pose of an object by keypoint matching.

To show the benefits of keypoint specific scoring, we perform three sets of measurements. First, as a baseline, we rank the nearest neighbor matches using the Hamming distance. Second, we rank only the nearest neighbor matches according to the scores of Eq. 3. Finally, we both pick the best match in the K near neighbor (KNN) list and rank these correspondences according to Eq. 3.

Table 1 lists the results obtained. The re-weighted nearest neighbor (RNN) values shows the improvement brought only by using the scores of Eq. 3. This leads to higher initial inlier ratios even though the final ratio at 500 matches stays nearly the same. The KNN curves show the additional improvement brought by searching beyond the nearest neighbor. In almost all cases, considering the KNNs leads to a higher inlier ratio over the best 500 keypoint matches.

For *Graffiti*, the weaker BRIEF performance compared to BRISK and FREAK is more than offset by exploiting the K near neighbors. For the other data sets, the BRIEF performance is either too low (*Boat*) or too high (*Bikes* and *Wall*) to lead to a real difference in performance.

For *Boat*, the baseline inlier ratios among the best 250 BRISK and FREAK correspondences are 44 and 45 %, respectively. The ten near neighbor inlier rates increase to 86 and 83 %, nearly doubling in each case. Such rates mean almost immediate convergence for PROSAC, requiring only 5 iterations to guarantee sampling of four inliers with 95 % probability, more than 20 times faster than the baseline.

4.3 Keypoint matching with locality sensitive hashing

For some vision tasks, the reference image collection is larger than a single image. In this case, the number of reference descriptors also increases, and therefore, it is impractical to match keypoints with brute-force descriptor search. To evaluate the performance of our approach in such a case, we concatenate the descriptors from all reference images from *Graffiti*, *Boat*, *Bikes*, *Wall*, plus the four other images. The total number of reference descriptors is around 8000. As in Sect. 4.2, we compute the inlier ratio curves, but this time we compute the list of ten near neighbors with LSH. We use the FLANN [17] implementation of LSH available in OpenCV with 12 tables, a key length of 20, and a multi-probe level of 2. This yields an LSH precision of 90 % (The nearest neighbor found by LSH is the same as found by the brute-force search nine out of ten times).

As Table 2 shows, the resulting inlier ratios are lower than those in Table 1 since the number of reference descriptors is eight times greater. However, our approach recovers matches that would have been lost if only the nearest neighbor had been used. The improvement (usually a factor of two) is more dramatic for *Boat* and *Graffiti*.

4.4 Computation time

For 1000 reference and 928 query keypoints, using a brute-force approach—that is when the descriptor distance to each reference keypoint is calculated—the total time for keypoint matching using only the nearest neighbor and Hamming distance is 4.0 milliseconds (using the POPCNT instruction on a 64-bit laptop CPU). At $K = 10$, our approach takes 4.2 milliseconds irrespective of the value of M. This means that the overhead to compute the scores of Eq. 3 for the top ten near neighbors is around 5 % for 1000 reference keypoints. The absolute overhead is negligible even for real-time applications, and at the worst case, it scales linearly with K; thus, it will be halved at $K = 5$ and doubled at $K = 20$.

5 Conclusion

We propose an approach that can be used in conjunction with the binary descriptors to search for keypoint matches beyond the nearest neighbor. Our approach strikes a balance between the keypoint specific representations and the generic descriptors. By using a two-step approach, we combine the advantages of both and improve the robustness of binary descriptors for real-time object detection applications.

References

1. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: Fast retina keypoint. In: Conference on Computer Vision and Pattern Recognition, pp. 510–517 (2012)
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51**(1), 117–122 (2008)
3. Balntas, V., Tang, L., Mikolajczyk, K.: Bold-binary online learned descriptor for efficient image matching. In: Conference on Computer Vision and Pattern Recognition (2015)
4. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York (2006)
5. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: computing a local binary descriptor very fast. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1281–1298 (2012)
6. Chum, O., Matas, J.: Matching with prosac—progressive sample consensus. In: Conference on Computer Vision and Pattern Recognition, pp. 220–226. San Diego, CA (2005)
7. Chum, O., Mikulik, A., Perdoch, M., Matas, J.: Total recall ii: Query expansion revisited. In: Conference on Computer Vision and Pattern Recognition, pp. 889–896 (2011)
8. Gupta, R., Mittal, A.: Smd: a locally stable monotonic change invariant feature descriptor. In: European Conference on Computer Vision, pp. 265–277 (2008)
9. Lepetit, V., Fua, P.: Keypoint recognition using randomized trees. IEEE Trans. Pattern Anal. Mach. Intell. **28**(9), 1465–1479 (2006)
10. Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: binary robust invariant scalable keypoints. In: International Conference on Computer Vision, pp. 2548–2555 (2011)
11. Levi, G., Hassner, T.: LATCH: learned arrangements of three patch codes. CoRR abs/1501.03719 (2015). http://www.openu.ac.il/home/hassner/projects/LATCH
12. Li, X., Larson, M., Hanjalic, A.: Pairwise geometric matching for large-scale object retrieval. In: Conference on Computer Vision and Pattern Recognition (2015)
13. Manning, C., Raghavan, P., Schütze, M.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
14. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2004)
15. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A comparison of affine region detectors. Int. J. Comput. Vision **65**(1/2), 43–72 (2005)
16. Morel, J.M., Yu, G.: Asift: a new framework for fully affine invariant image comparison. SIAM J. Imaging Sci. **2**(2), 438–469 (2009)
17. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 2227–2240 (2014)
18. Oszust, M.: An optimisation approach to the design of a fast, compact and distinctive binary descriptor. Signal Image Video Process. (2016). doi:10.1007/s11760-016-0907-4
19. Ozuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast keypoint recognition using random ferns. IEEE Trans. Pattern Anal. Mach. Intell. **32**(3), 448–461 (2010)
20. Proença, H.: Performance evaluation of keypoint detection and matching techniques on grayscale data. SIViP **9**(5), 1009–1019 (2015)
21. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: an efficient alternative to sift or surf. In: International Conference on Computer Vision, pp. 2564–2571 (2011)
22. Trzcinski, T., Christoudias, M., Lepetit, V.: Learning image descriptors with boosting. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 597–610 (2015)