KEYPOINT MATCHING BASED ON DESCRIPTOR STATISTICS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Furkan Eren UZYILDIRIM

July 2016
İZMİR

We approve the thesis of **Furkan Eren UZYILDIRIM**

Examining Committee Members:

Prof. Dr. A. Aydın ALATAN
Department of Electrical and Electronics Engineering, Middle East Technical University

---

Asst. Prof. Dr. Yalın BAŞTANLAR
Department of Computer Engineering, İzmir Institute of Technology

---

Asst. Prof. Dr. Mustafa ÖZUYSAL
Department of Computer Engineering, İzmir Institute of Technology

22 July 2016

Asst. Prof. Dr. Mustafa ÖZUYSAL
Supervisor, Department of Computer Engineering
İzmir Institute of Technology

Assoc. Prof. Dr. Yusuf Murat ERTEN
Head of the Department of
Computer Engineering

Prof. Dr. Bilge KARAÇALI
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my supervisor, Mustafa ÖZUYSAL, who advised and encouraged me all the way during this thesis work. I am grateful for his motivation, patience, continuous support and for sharing his immense knowledge with me. I would like to also thank him for providing lab environment which always motivate me during my thesis work. Being a member of Visual Intelligence Research Group makes me happy and provides feeling of being in the correct place for research studies.

I would like to thank my friends for all the support and their motivation during busy and tiring times.

Finally, I would like to express my infinite gratitude to my parents for their unconditional love and endless support without any expectations. I dedicated this thesis work to them.

This thesis is supported by The Scientific and Technical Research Council of Turkey (TUBITAK) under the grant 113E496.

ABSTRACT

KEYPOINT MATCHING BASED ON DESCRIPTOR STATISTICS

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. In this thesis, 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 bit variations collected for each keypoint individually in an off-line training phase. This is similar in spirit to approaches that learn a patch specific keypoint representation. Unlike these approaches, we limit the use of a keypoint specific score only 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 collections.

ÖZET

BETİMLEYİCİ İSTATİSTİKLERİ İLE ANAHTAR NOKTALARIN EŞLEŞTİRİLMESİ

Gerçek zamanlı uygulamalarda, ikili betimleyiciler anahtar noktaların eşleştirilmesinde sıklıkla kullanılmaktadır. İkili betimleyicilerin ayrık yapıları doğru eşleşmelerin sayısında düşüklüğe sebep olmaktadır. Bu tez çalışmasında, doğru eşleşme en yakın komşu yerine en iyi K komşuyu içeren bir liste üzerinde aranarak ikili betimleyicilerin performansında iyileştirme amaçlanmıştır. Bu liste üzerinde doğru eşleşmeyi seçebilmek için, her anahtar noktanın ikili betimleyicisinin bit değişimlerinin istatistikleri çevrimdışı eğitim aşamasında çıkarılmıştır. Bu yöntem, anahtar noktaları etrafındaki belirli bir imge alanıyla temsil eden anahtar nokta özgü yaklaşımlara benzerlik göstermektedir. Bu yaklaşımlardan farklı olarak, anahtar noktaya özgü skor sadece en yakın K komşuyu içeren listeyi sıralamak için kullanılmıştır. Bu liste yaklaşık en yakın komşuyu bulma algoritmalarıyla birlikte hızlı ve verimli olarak hesaplanabileceği için, geliştirilen yöntem büyük betimleyici kümelerine ölçeklendirilebilir.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

Many of the Computer Vision applications include any of the processes of image matching and identification, object recognition and tracking, image retrieval, and so on [2, 21, 13, 11]. These tasks extract distinctive features in an image as the first step. Keypoints are point-based features which are able to be detected in a computationally efficient way. Generally, they are only composed of location, scale and orientation information. This cannot be enough for establishing correspondences between images which is the crucial part of the real-time vision systems. This problem can be solved by representing each keypoint with the patch surrounding it which is invariant to deformations such as viewpoint and illumination changes. This representation is called a descriptor which is usually a high dimensional binary or floating-point vector.

Thanks to important developments in mobile platforms, computer vision applications which are worked on devices with high processing power and quite enough memory can be adapted to mobile devices. These devices have limited memory and low processing power. Transmission of large-sized data through mobile network is also not feasible. In order to deal with these concerns, several binary descriptors have been introduced in the recent years as an alternative to floating-point ones which require high-performance processors for execution and high amount of memory for storage.

Binary descriptors are simply bit-streams. Correspondences between the points on the variously deformed images can easily be obtained with binary descriptors based on the Hamming distance which is computed by bit-wise operations. The aim of binary descriptors is to obtain the small distance between the keypoints in different images that are belong to the same scene point. This requires to have as few as possible number of different bits between the descriptors of these keypoints. Each bit of the binary descriptor is computed based on intensity comparison of two pixels on the given patch. Points that are selected for the computation of descriptor bits are the same for all patches specifically to the descriptor type. However, these points represent different textures for different views of the keypoint so that the same point pair might not give the same bit value for the resultant descriptor. Moreover, with various deformations, the number of different bits

and also the invariant bits are dependent on the texture in the patch which is unique for each keypoint. These variations adversely affect the performance of binary descriptors and make establishing correct matches between images difficult with them.

In this thesis, we focus on improving the robustness of the keypoint matching with binary descriptors by taking into account the bit variations for each keypoint under deformations besides the distance information that can be obtained by highly efficient computation.

1.1. Motivation

As mentioned previously, keypoint matching can be achieved with binary descriptors based on the Hamming distance between descriptor vectors. In real-time vision systems, the reference keypoint whose descriptor gives the *smallest* distance to the descriptor of the query keypoint is assigned as match. In other words, *the nearest neighbour* is searched for the query descriptor over the set of reference descriptors. This way of matching leads to obtain incorrect matches even the distance for the correct match of the given query descriptor is only one more than the distance given by the nearest neighbour. However, when we observe the top K near neighbours by ranking the matches based on the Hamming distance between their descriptors, we notice that this set of matches is more likely to contain the correct match. Figure 1.1 illustrates the existence of correct matches beyond the nearest neighbor. It shows the number of correct matches between BRIEF [7] descriptors that fall into the first ten near neighbors. Although most of the BRIEF matches obtained by the nearest neighbour matching are wrong, the correct ones are not very far away in the near neighbor list. The recognition rate over the first ten near neighbors is 61 percent and over only the nearest neighbor is 19 percent. The matches beyond the nearest neighbor are only reachable if there is a way of correctly reranking the matches to bring the correct one to the top.

Under different viewpoint changes, variations in descriptor bits occur due to the change in texture on the patch in which descriptor computation is done. Since each keypoint has its own texture, the effect of deformations is specific for the descriptor of each keypoint which is discussed in Chapter 4 in detail. In order to handle this, instead of representing each keypoint with a single reference patch, each can be represented by multiple patches by simulating various viewpoints. Binary descriptors which are computed

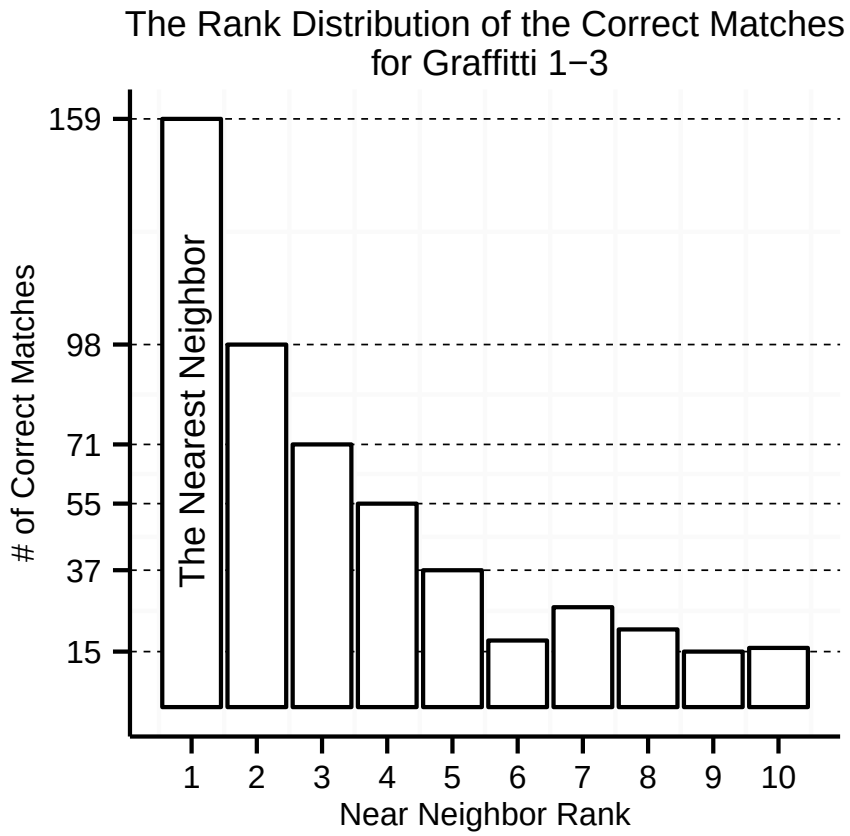


Figure 1.1. The near neighbor rank distribution of the correct matches between BRIEF descriptors extracted from the first and the third images of the Graffiti data set. The nearest neighbor obtained by ranking matches according to Hamming distance captures only 159 of the 848 possible correspondences. Meanwhile, the first ten near neighbors include 517 correct matches, a potential improvement by a factor of more than three.

from this set of patches provide to obtain overall bit variations under various deformations specifically to the each keypoint. These observations give intuition to us for reordering the list of top K matches based on the keypoint-specific representations.

1.2. Thesis Goals and Contributions

This thesis aims to develop an approach to enlarge the matching range of the binary descriptors. We require to use generic descriptors to make the approach scalable for

large descriptor sets and keypoint specific representations to make it robust for under various viewpoints. This requirement is fulfilled by extracting the statistics for the descriptor variations of each keypoint in an offline training phase to reorder match hypotheses in the list of K near neighbours at run-time.

Our main contributions in this thesis can be summarized as follows:

- We develop a two-step approach for keypoint matching by combining the generic keypoint descriptors and keypoint specific representations.
- 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 develop a training procedure that covers various deformations which provides to exploit the texture characteristics around individual keypoints that yields a more robust and distinctive description.
- We develop the keypoint specific and probabilistic score for reranking the keypoints within the list of K near neighbours.
- We demonstrate that, despite learning a separate representation for each individual keypoint, our approach does not require a brute-force search in a large descriptor set when coupled with the LSH [3] approach to compute the list of K near neighbors.
- We also show that our approach is relatively descriptor independent and it extends the matching range of several binary descriptors beyond the nearest neighbor: BRIEF [7], ORB [29], BRISK [16], FREAK [1], and LATCH [17], which has recently been shown to outperform state-of-the-art binary descriptors.

1.3. Outline of Thesis

This thesis is organized as follows. The next chapter provides a literature overview. Chapter 3 gives a background information about keypoints, descriptors, and their applications. Chapter 4 includes the detailed research for the stability of keypoints and binary descriptors under various viewpoint changes. In Chapter 5, we propose an approach for keypoint matching with binary descriptors. Finally, Chapter 6 provides final remarks and discusses future research.

CHAPTER 2

RELATED WORK

As mentioned previously, representing features extracted from the images with a distinctive and robust descriptors is one of the essentials for most of the vision systems. The correspondences between features from different images are found by feature descriptors. Keypoints which are point-like features are represented with the descriptor vectors computed from the patch surrounding them.

Keypoints can be represented by floating-point descriptor vectors such as SIFT [19]. It computes the keypoint descriptors based on the histogram of gradients in order to make the descriptor rotation invariant. With SIFT, keypoint matching is done based on the Euclidean distance between the descriptor vectors. It performs well in most of the matching tasks and widely used in computer vision applications such as image retrieval [26] and panoramic image stitching [6]. Although some extensions for SIFT are introduced [5, 22, 14] for more efficient computation or dimensionality reduction, such type of floating-point descriptors suffer from their requirement for high processing power and high amount of memory to store them. However, Chandrasekhar et. al. [8] proposed an approach that compress the floating-point descriptors [19, 5] to store and transmit them efficiently. They encode feature descriptors by applying the invertible linear transform since the descriptor is decompressed by the decoder at run-time. They obtain image matching and retrieval performance that is close to the conventional floating-point representations by encoding each feature dimension approximately with 2 bits.

In order to provide efficiently computed and less amount of sized keypoint representations, several binary descriptors have been proposed [7, 29, 16, 1, 17] in recent years. These are efficiently computed by intensity comparisons of two pixels on the patch centred by the keypoint and the resultant descriptor vector is a simply bit string. This simplicity and compactness of the binary descriptors make them appropriate for real-time vision systems especially work on mobile devices which have low amount of memory and low-performance processors. Visual SLAM applications such as [30] aim at keeping track of the camera location and updating the 3D map of the scene points simultaneously. To construct the 3D map of the scene points, keypoint correspondences are established

between two frames. These correspondences are found by binary descriptors in such applications.

Unfortunately, binary descriptors cannot reach the matching performance provided by floating-point ones due to some properties of their nature which is discussed in Chapter 4 in detail. In order to improve the performance of the binary descriptors, some approaches are proposed so that patch specific binary descriptors are computed which provides to represent each keypoint in a more distinctive and robust way. In our approach, this is provided by computing keypoint spesific score obtained from the bit variations under different views of the patch surrounding the keypoint.

There are a few existing approaches that using binary feature descriptors by learning and using keypoint specific representations [4, 12, 15, 27]. Mikolasjczyk et. al. [4] proposed a keypoint specific matching approach by computing a binary descriptor specifically adapted to a given image patch. They learn a combined set of generic features and a locally optimized binary mask that picks the best features depending on the image patch. The details of this approach is given in the Section 3.4.2.

Mittal et. al. [12] proposed selecting a patch specific set of binary features that are robust to changes in intensity levels as well as small translation and rotations. They split each patch into two regions by setting a certain intensity level as a threshold. Points that have intensity level that is bigger than the threshold form a region while others are included into the other region. To determine binary tests, they select points from different regions and points which are far away from the boundary of their own region. For different intensity level thresholds, they obtain binary tests with the same procedure. To compute the feature descriptor, they eliminate point pairs which possibly are not invariant to illumination changes or some spatial degradations such as translations and rotations. Remaining ones form the feature descriptor of the corresponding patch. They compute a score between two feature descriptors for matching. This score is obtained by testing the stability of the each binary test of the query descriptor in the patch of each reference descriptor by concerning order flips.

Both [4] and [12] 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.

Lepetit et. al. [15] proposed a keypoint classification approach by using randomized trees. In an off-line training phase, they select robust keypoints for perspective changes and assign each of it as a class with the set of patches surrounding them un-

der different viewpoints. They create the set of randomized trees as an classifier so that each node represents the single binary test. For each keypoint , the node checks whether the intensity level of the first pixel of the binary test is smaller than the second one or not. If it is, the keypoint is then assigned to the left child, otherwise to the right child. These node tests are applied until it reaches a leaf. Each leaf node of the trees stores the number of patches of each class and also the total number of patches reaches to it as the posterior probabilities. At runtime, the same node tests with the ones in the training are applied for each query keypoint in each of the trees. For each class, the probability of reaching leaf nodes that are hit by the query keypoint is then computed. Finally, the query keypoint is assigned to the class that gives the highest probability.

Ozuysal et. al. [27] proposed keypoint classification approach by training keypoint specific classifiers. They select a subset of the keypoints extracted on the reference images by applying affine deformations to the image in order to determine stable keypoints. Each selected keypoint is assigned as class. In order to obtain the training set of the each class, multiple views of the patch centred by the keypoint are generated under different distortions. They select the binary features randomly from the patch at the beginning of the training phase. These features are split into the groups which are independent from each other and statistics for each are computed during training. For the given patch surrounding a keypoint, the same binary features with the ones used for training are selected on the patch and they are split into the groups by the same way used in training. For each class, the probability of observing the given keypoint is calculated based on independent groups of binary features. The keypoint is then assigned to the class that gives the highest probability.

Both [15] and [27] compute a probability distribution over all reference keypoints at run-time. Moreover, they require large amounts of memory per keypoint to store the learned probabilities. As a result, they are not suitable for large keypoint data sets.

The combination of a generic query step to create a short list of candidates followed by candidate specific filtering is quite common in the image retrieval literature [10, 18] due to the large size of the image collections. For image retrieval, the filtering in the second stage depends on the geometric consistency of keypoint correspondences between the query image and the candidate query results. Our approach has a very similar pipeline where the geometry check is replaced by a probabilistic observation probability.

CHAPTER 3

RESEARCH BACKGROUND

3.1. Keypoint as an Image Feature

In most of the computer-vision systems, the first step is extracting the features of an image. These features should be distinctive and stable in order to make an image identifiable under distortions such as viewpoint and illumination changes. An image feature can be a single pixel such as corners or a group of contiguous pixels like edges. Keypoints are point-like features on an image. A keypoint can be a corner or any other pixel which has a strong intensity difference with its neighbour points. Keypoints are highly preferable image features for most of the real-time applications since they are fast to extract and robust to various deformations which is shown in Chapter 4.

3.2. Keypoint Detection

As it is discussed in Section 3.1, keypoints are distinguishing features on an image. Keypoint detection is the process of extracting the keypoints and its main purpose is to localize characteristic features of an image under various distortions.

There are several keypoint detection algorithms. Generally, keypoint detectors determine keypoints based on intensity comparisons. Most of the algorithms have two-step process. In the first step, all points that satisfy the condition of being a keypoint are determined. This set of features is then pruned in order to keep keypoints which have more potential to be robust to deformations and to represent the same scene points on an image in different views.

In the following, we describe the one of the computationally efficient keypoint detection algorithms which is called Features from Accelerated Test (FAST) [28].

3.2.1. FAST Keypoint Detector

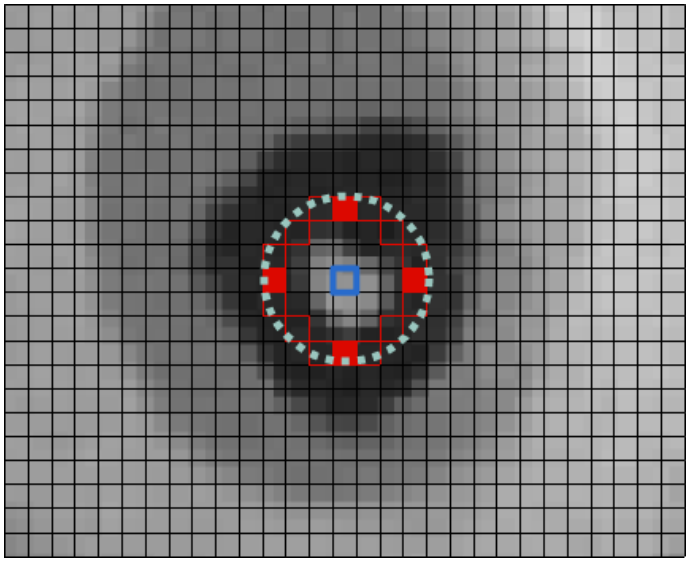


Figure 3.1. Keypoint extraction with FAST. For each pixel p in an image, FAST makes comparison with 16 pixels locate on the circle with a radius of 3, centred by the point p based on their intensity levels. The intensity level of point p should be bigger or smaller than the 12 consecutive pixels with a preset intensity level threshold t on the circle in order to point p become a keypoint candidate. FAST first compares pixels in the compass directions of the point p which are illustrated with filled coloured pixels. Three of them should have intensity level that is either smaller or bigger than the point p's with a threshold t in order to point p is assigned as a keypoint candidate.

As in the most of the keypoint detectors, FAST initially searches keypoint candidates through all of the pixels in an image. It fits a grid on an image so that each cell of a grid belongs to an individual pixel. In order to decide whether a point p has potential to be a keypoint or not, FAST compares it with 16 pixels based on intensities. These pixels reside on the circle with a radius of 3 centred by the p as it is illustrated in Figure 3.1. The intensity comparisons are done with preset intensity level threshold t. If the intensity level of the center pixel I_p is either bigger or smaller than the intensity level of 12 consecutive pixels on the circle by the value of t, then p is assigned as keypoint candidate. For faster implementation, FAST first checks the pixels in the circle reside in the four compass directions of the point p which are shown with filled color in Figure 3.1. If three of them does not satisfy the condition based on intensity levels then point p is rejected as a possible keypoint.

FAST can assign two or more adjacent pixels as keypoint candidates. Assigning these successive candidates as keypoints leads to problems in matching between different views of the scenes on the image. So, in order to select the more distinctive features over the set of keypoint candidates, non-maxima suppression is applied. For each candidate, a score is computed. This score is the sum of the absolute difference between the intensity level of the keypoint candidate and each of contiguous 12 pixels. Candidates which have an adjacent keypoint candidate with higher score are eliminated. Remaining ones are then assigned as keypoints.

Beginning the comparison with the pixels of compass directions brings computational efficiency. Most of the points that cannot be a keypoint are excluded in a fast way by making comparison only with 4 pixels of 16. On the other hand, there are some rejected points whose pixels in compass directions satisfy the condition but one of the other consecutive pixels on the circle makes the point p be rejected. So, until this pixel on the circle is taken into account the algorithm makes unnecessary comparisons which leads to have more execution time. In order to accelerate the speed of the computation, FAST uses machine learning approaches to obtain the order that 16 pixels on the circle are selected for comparison.

3.3. Keypoint Description

As it is stated in Section 3.2, keypoint detection is finding distinctive features of an image that represents the different views of the scenes on the image. After keypoint detection, keypoints are represented with their locations and some of the detectors also provides scale and orientation information.

In real-time applications, the location information cannot be enough for deciding whether two points in different images are belong to the same scene point or not. So, each keypoint is represented with its descriptor vector that is computed from the patch surrounding it. In other words, each region around keypoint locations is converted into a descriptor vector by targeting to make it invariant under various distortions.

Descriptor vector of a keypoint can be computed based on histogram of gradients as in [19]. For computation, the gradients of each pixel in the patch need to be computed. Such descriptor computation leads to have high processing time. It also results in consuming high level of the memory since each descriptor vector consists of 128 floats. So, it

cannot be feasible for applications that have real-time constraints and need to store large number of descriptors.

Binary descriptors are efficient alternatives for floating point ones like [19, 5]. They are more computationally efficient and require less amount of memory to be stored. Real-time systems such as mobile applications has limited size of memory and low computational power. So, representing keypoints with binary descriptor vectors is advantageous especially for using resources efficiently in real-time vision applications.

In the following, computation of binary descriptor vectors is described and also several binary descriptors are introduced.

3.3.1. Binary Descriptor Computation

The binary descriptor vector is computed from the patch P with a size of SxS around the keypoint. The resultant vector is a binary string in which each bit is computed from the pair of pixels that are selected from the patch P. These pairs are also called binary tests. In each binary test, intensity level comparison is done. If the intensity of the first point in the pair is higher than the second one's then this test gives 1 to the descriptor vector otherwise it gives 0 as it is illustrated in Figure 3.2. To build a binary descriptor vector of length N, the same number of pair of pixels are selected in the patch P. Generally, N is equal to 256 or 512.

There are several binary descriptors which differ from each other in the following properties [33]:

- Sampling Pattern: The method for selecting points in the patch surrounding the keypoint.
- Orientation Compensation: The method for estimating orientation of the keypoint.
- Sampling Pairs: The method for determining pairs of pixels from the set of points obtained by the sampling pattern.

BRIEF [7] is the first binary descriptor that was introduced. It does not have a sampling pattern and orientation estimation mechanism. It determines binary tests that are the pairs of pixels in the patch randomly by selecting any of the point in the patch. In order to reduce the effect of the noise, BRIEF smooths the patches before the computation.

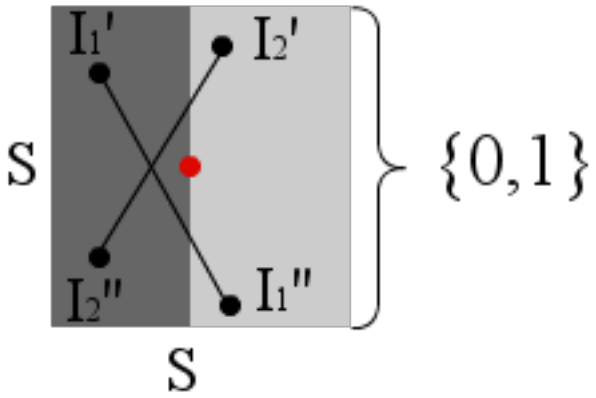


Figure 3.2. Binary descriptor vector computation with two binary tests in the patch with a size of SxS centred by the keypoint . For the first one, the intensity level of the first point I_1' is lower than than the intensity level of the second point I_1". So, this test gives 0 to the resultant vector. In second binary test, the first point has higher intensity level than the second one(I_2' $>$ I_2") which results to give 1 to the descriptor vector.

ORB [29] does not have a sampling pattern like BRIEF but it has an orientation estimation mechanism. It estimates the orientation of the patch according to local first order moments within the patch and rotate each binary test by the corresponding orientation in order to be rotation invariant. ORB uses unsupervised learning in order to determine sampling pairs from the set of sampling points. By running each possible binary test on all training patches, sampling pairs which have few correlation with the other binary tests and ones that have high variance with the other tests are selected in order to make the feature vector discriminative.

BRISK [16] has a sampling pattern which is composed of concentric rings. For each sampling point, Gaussian smoothing is applied to a patch around it and the size of the standard deviation increases from inner to outer rings. From the points on the sampling pattern, BRISK obtains sampling pairs. These sampling pairs are separated into two groups according to spatial distance between pixels of a sampling pair. Short pairs which have a distance between pixels that is smaller than a certain spatial distance threshold are used for building the descriptor vector. Long pairs which have a distance between pixels that is higher than a threshold are used for orientation estimation. In order to estimate the orientation of the patch BRISK uses local gradients between the long sampling pairs.

FREAK [1] has a circular sampling pattern like BRISK, but unlike it, the density

of points in inner rings is higher than the outer ones. As in BRISK, each sampling point is smoothed with a Gaussian kernel with a size which increases from the points in inner rings to ones in outer rings. In order to determine the sampling pairs over sampling points, FREAK applies the same approach with ORB which uses unsupervised learning to obtain sampling pairs which have few correlation and high variance among them. In order to estimate the orientation of the patch, FREAK uses the same approach with the BRISK but instead of using all sampling pairs which have large distance between corresponding pixels, it uses 45 symmetric sampling pairs.

The simple representation for the sampling patterns of the binary descriptors explained in the above is illustrated in Figure 3.3.

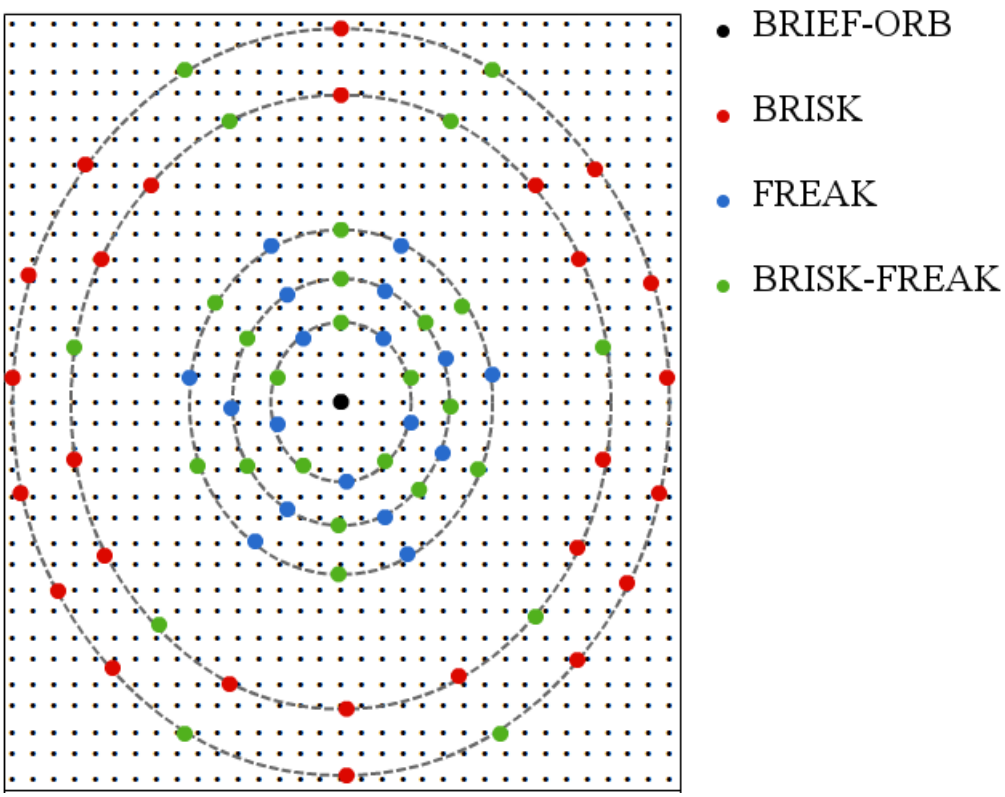


Figure 3.3. Simple representation for the sampling patterns of binary descriptors. BRIEF and ORB have no any sampling pattern so that each point(black points) on the patch surrounding the keypoint can be used to obtain binary tests. BRISK and FREAK have similar sampling pattern which is composed out of concentric rings around the keypoint. The difference is that unlike to BRISK(uses red and green points to obtain binary tests), FREAK gets more samples(blue and green points) from the rings near the keypoint than the samples on outer rings.

LATCH [17] has a different mechanism than the descriptors which are explained in the above. Instead of comparing sampling pairs which are composed of pixel pairs, LATCH selects patch triplets with a small size and make comparisons on these. For each triplet, the sum-of-squared differences are computed from the one of the patches to each of other two patches and based on these two distances the output bit value is determined. In order to select an optimal set of triplets, supervised learning is used unlike to ORB.

3.4. Keypoint Matching with Binary Descriptors

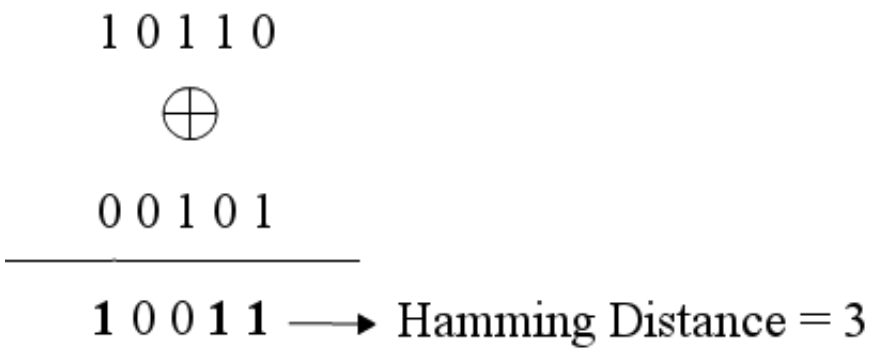


Figure 3.4. Hamming distance computation for two binary strings with a length of 5 bits.

Keypoint matching is the acquisition of correspondences between the images. Its aim is correctly match the keypoints which are belong to the same scene point in different images. In real-time applications, keypoints are represented with their descriptor vectors and these are generally binary vectors as it is discussed in Section 3.3. So, binary descriptors are widely used for keypoint matching.

The image that is given to the system for identification or matching is called *query image* and its descriptors are named as *query descriptors*. The set of descriptors in which the match of each query descriptor is searched is called *reference descriptors* which are computed from the *reference images* which do not have any deformation.

Binary descriptor vectors are bit-streams with a length of 256 or 512 in general. Finding the match for the query descriptor over reference descriptors is done based on the distance between binary descriptor vectors. The metric that is used to measure this distance is called *Hamming distance*. It is computed in a computationally efficient way.

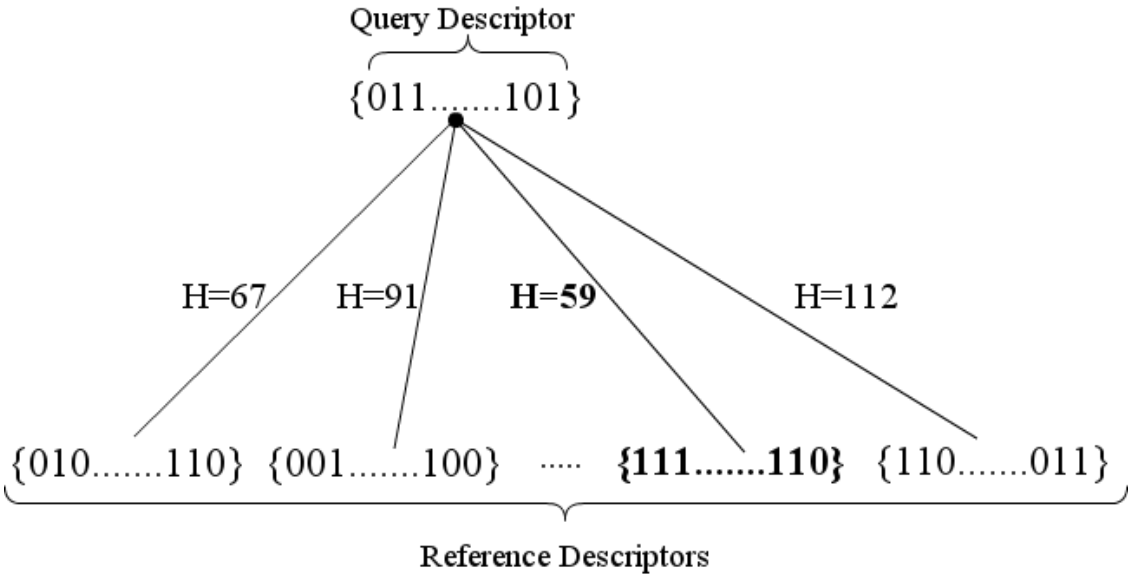


Figure 3.5. Keypoint matching with 256-bit length binary descriptors. For the query descriptor, the Hamming distance H is computed with each of the reference descriptors. Then the query descriptor is matched with the reference descriptor that gives the smallest Hamming distance($H = 59$ in the figure).

Firstly, exclusive or (XOR) of the two binary strings is taken and the number of 1s in the final bit-stream gives the Hamming distance as it is illustrated in Figure 3.4 for 5-bit length binary strings. The reference descriptor that gives the smallest Hamming distance is assigned as match to the query descriptor as it is shown in Figure 3.5.

Searching the match for the query descriptor over the reference descriptor set by computing the distance with each reference descriptor like in Figure 3.5 is not feasible for some vision applications. This brute-force search approach cannot be applicable for the vision tasks that have large reference image collection. So, approximate nearest neighbour methods are used to find the match of the query descriptor and this approximate solution is often enough and can be found in an efficient way .

3.4.1. Keypoint Matching with Approximate Nearest Neighbour Algorithms

In some vision systems, the set of reference descriptors is large and computing the distance from the query keypoint to each reference keypoint is not practical. For such cases, the approximate nearest neighbour(ANN) techniques are used in order to limit the exact nearest neighbour search to the subset of reference descriptors.

3.4.1.1. Locality Sensitive Hashing

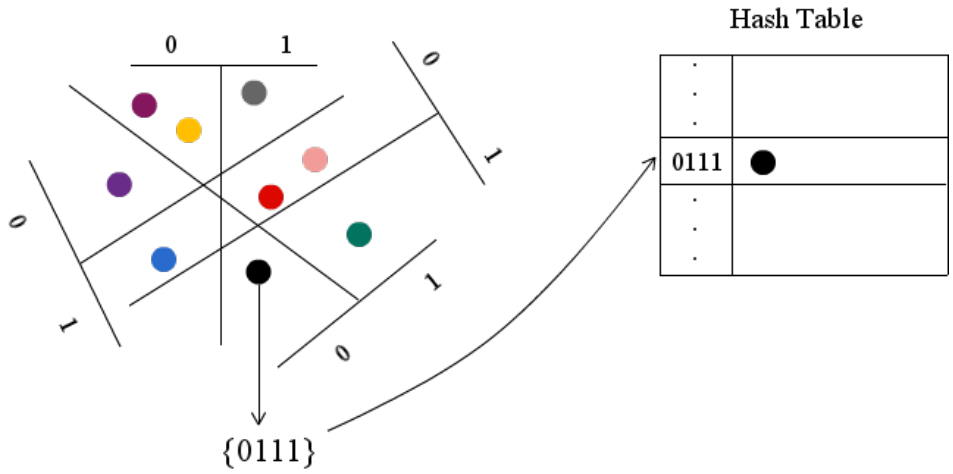


Figure 3.6. Example for LSH algorithm with hash key with a length of 4. Here, hash functions are hyperplanes that separates the n dimensional space into two and either 0 or 1 is assigned as corresponding hash value. The n dimensional data is projected into the 4-bit vector and mapped to the corresponding bucket in a hash table. For each of the N hash tables, data is given to the such set of hash functions and hash key is computed. Then, it is mapped to the one of the buckets of the corresponding table according to computed hash key.

Locality Sensitive Hashing(LSH) [3] is one of the ANN algorithms that is project the data into low dimensional binary space. Each data point is mapped to a k-bit vector which is called as $hash\ key$. LSH algorithm uses N hash tables in which each table is represented with a different set of hash functions S. Each S is obtained by concatenating

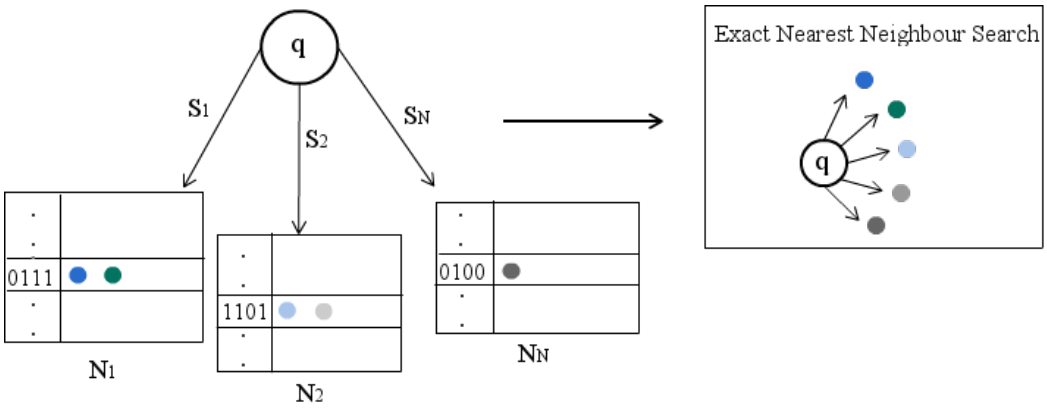


Figure 3.7. The query data q is given to the each hash function S specific for each hash table. Then, the exact nearest neighbour search is done only with data which are in the same bucket with hashed q in each hash table.

k hash functions. The output of each hash function is a single bit so that after data is given to each hash function, it is mapped to the one of the buckets in the hash table as it is illustrated in Figure 3.6.

Given a query data, LSH iterates over the N hash tables. For each hash table, it retrieves the data that are hashed into the same bucket as query and exact nearest neighbour search is only done for these data as it is shown in Figure 3.7.

3.4.2. Keypoint Matching with Keypoint Specific Representations

As it is stated in Chapter 2, there are a few keypoint specific representations that use binary feature descriptors for keypoint matching. Mikolasjczyk et.al. [4] proposed such approach which computes a binary descriptor specifically adapted to a given patch.

In an offline phase, they obtain all possible binary tests (point pairs) on the patches which are obtained from the training patch dataset in order to select most discriminative ones. They construct a matrix that each row represents an individual patch and each column belongs to the feature computed from corresponding binary test. In order to select uncorrelated features which are binary tests with high variance, they sort columns of matrix according to occurrence of 1s since this value is directly proportional to variance and assign top 512 binary tests as combined set of generic features.

At run-time, patch around each keypoint is represented with the set of binary tests

learned in an offline phase and locally optimized binary mask to determine the best features depending on the patch. To calculate the binary mask, they apply affine transformations to all selected binary tests on corresponding patch. If the given feature by the binary test remains the same after these transformations then its value on the mask is assigned as 1 otherwise it is assigned as 0. They claim that a few affine transformations are enough to determine whether a binary test is stable or not. With binary mask, only robust binary tests which are specific to each keypoint are taken into account to obtain the final symmetric Hamming distance between the query descriptor and each of the reference descriptors. The matching of two keypoints with their descriptors is done by the symmetric Hamming distance which is computed as follows:

$$H(Q, R_i) = \frac{1}{N_q} m_q \wedge g_q \oplus g_r + \frac{1}{N_r} m_r \wedge g_q \oplus g_r \ , \tag{3.1}$$

where Q is the query descriptor and R^i is the descriptor for the reference keypoint i. m_q and m_r are binary masks corresponding to the query and the reference descriptor respectively. N_q and N_r are the number of 1's in m_q and m_r respectively. g_q and g_r are binary descriptors of the query and the reference keypoint respectively which are computed from binary tests learned in an offline phase.

CHAPTER 4

KEYPOINT AND BINARY DESCRIPTOR STABILITY

4.1. Introduction

Keypoints are distinctive features on the image. They have potential to represent the same scene points in different images. Keypoint detection is the process of extracting keypoints and it aims to localize distinctive features that belong to the image also under various perspective changes or different environmental conditions. The main criterion for evaluating quality of a keypoint detector is called *repeatability*, which is the ability of a detector to locate keypoints of the same scene point in different images. Environment conditions and camera positions in which repeatability is under acceptable values lead to an insufficient number of detected keypoints. This results failure in object detection and tracking.

Keypoint matching is finding keypoints that belong to the same scene point on an image pair. If the geometrical relation between two images is known then keypoint locations can be directly used for matching. In real applications, transformation details between images is unknown. In such cases, matching can be achieved by comparing descriptor vectors of keypoints computed from the patch surrounding them. Generally, these descriptors are binary in real-time applications. Two keypoints are matched if and only if the distance between their descriptors are small. The binary descriptors are particularly sensitive to larger changes in viewpoint and scale. Variations on binary descriptors under perspective transformations makes highly accurate keypoint matching difficult.

In this chapter, we measure repeatability of the keypoint detector for geometric deformations and also we quantify variations of binary descriptors under these deformations separably for each keypoint.

4.2. Approach

In contrast to 3D object space, any image pair of the planar surface has a simple geometrical relation itself and this relation can be modelled by a homography matrix. Scenes on an image can be sampled from different viewpoints by computing various homographies and applying these to a reference image of the surface.

Stability of keypoints is evaluated by their detectability in different views of the scenes on the image. For this evaluation, we measure the repeatability which is the number of extractable keypoints in an another view of the scene on the image. In order to

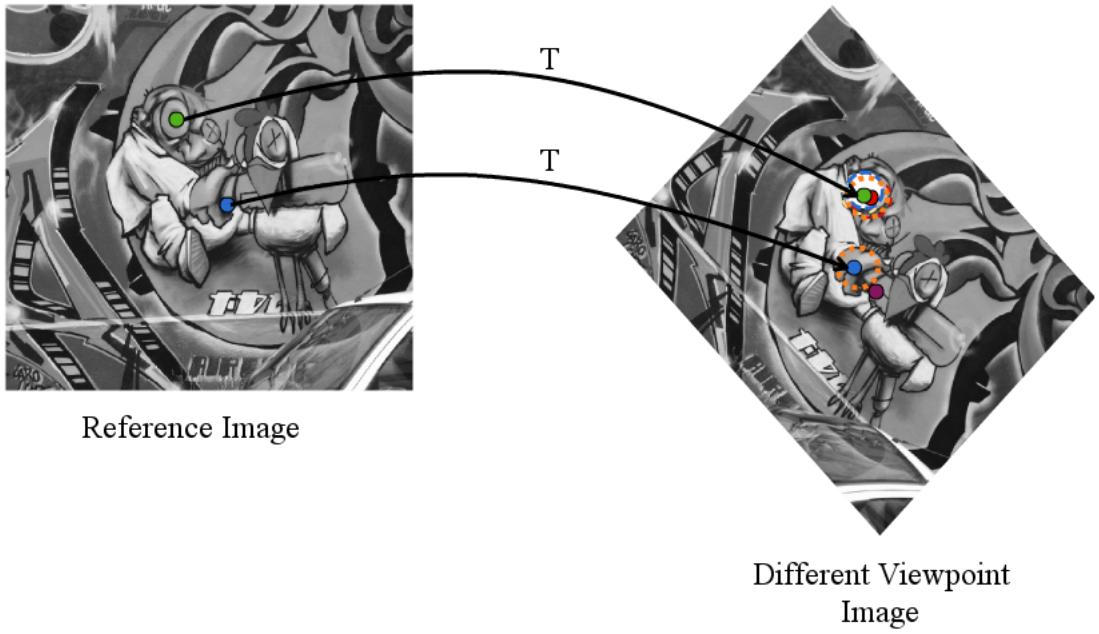


Figure 4.1. Repeatability decision for the keypoints. Each keypoint on the reference image is warped to different viewpoint image by the corresponding ground-truth transformation matrix. If Euclidean distance between locations of the transformed keypoint and any of the keypoints on the sampled image is smaller than the 2 pixels then the reference keypoint is accepted as repeatable on the different viewpoint image.

decide whether a keypoint repeats or not, each keypoint in the reference image is transformed to images sampled from different viewpoints by using the ground truth transformation matrices. Euclidean distance is then computed between each reference keypoint k_i and all keypoints on the different viewpoint image according to their locations. If any of the distances is smaller than 2 pixels then it is accepted that keypoint k_i repeats on the

sampled image as it is illustrated in Figure 4.1. Distance of 2 pixels can be used by many object detection and image matching methods. Smaller distance thresholds are more restrictive and bigger ones can not be enough for accurate detection of object location.

We compute the repeatability for each different viewpoint image by formula given in the below:

$$R_i = \% \frac{N_i}{N_s} \tag{4.1}$$

where R_i is the repeatability value of ith sampled image from different viewpoint, N_i is the number of the keypoints that repeats in the ith sampled image and N_s is the number of keypoints that are detected in the source image.

In order to obtain scenes on the image from different viewpoints, we generate synthetic images by using affine deformation model proposed by [24] which computes affine homographies by sampling different camera position parameters as illustrated in Figure 4.2:

- Scale (λ in [24]) is the zoom parameter,
- In-plane rotation angle (ψ in [24]) represents rotation around an axis that is perpendicular to the object plane,
- Tilt amount (θ in [24]) is the angle between the normal of the image plane and the optical axis of the camera ,
- Tilt angle (ϕ in [24]) is the angle in which tilt is applied.

The affine deformation(homography) matrix A can be obtained by using deformation parameters introduced above by the following equation:

$$A = \lambda \begin{bmatrix} cos\psi & sin\psi \\ -sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} cos\phi & sin\phi \\ -sin\phi & cos\phi \end{bmatrix} \tag{4.2}$$

where t refers to tilt and represents the deformation degree of the image that is calculated as follows:

$$t = 1/cos(\theta) \tag{4.3}$$

For each synthetic image, size is computed separately in order to transform each pixel of the source image to the deformed image correctly otherwise some pixels will fall outside of the synthetic image. Size of the synthetically generated image equals to the

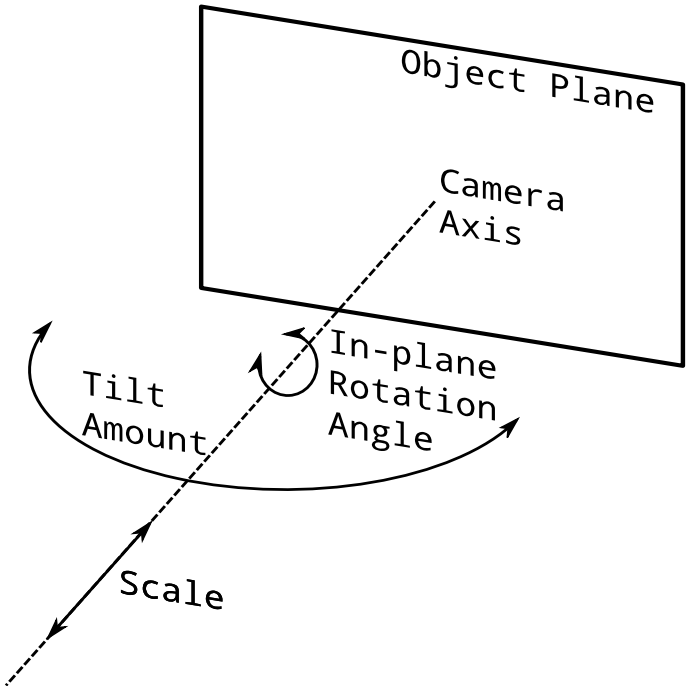


Figure 4.2. Affine deformation parameters sampled in the experiments. Each parameter corresponds to a particular motion of the camera with respect to object plane.

size of the smallest rectangle that encapsulates points which are obtained after corners of the source image are warped with transformation matrix into the deformed image.

Transformation matrix T is then calculated by multiplying A with rotation and inverse rotation matrices in order to transform each point on the image correctly as follows:

$$T = \begin{bmatrix} 0 & 0 & \frac{d_w+1}{2} \\ 0 & 0 & \frac{d_h+1}{2} \\ 0 & 0 & 1 \end{bmatrix} A \begin{bmatrix} 0 & 0 & \frac{-s_w}{2} \\ 0 & 0 & \frac{-s_h}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

where d_w and d_h represents width and height of the deformed image respectively and in order of s_w and s_h represents width and height of the source image.

In order to transform keypoint p to the deformed images, coordinates of the p are multiplied by the transformation matrix T. Since the size of T is 3x3, 2D location of

keypoints is represented by homogeneous coordinates as a 3D vector in order to make it appropriate for matrix multiplication as follows:

$$p = (x, y) \longrightarrow p = (x_1, y_1, z_1) \tag{4.5}$$

where $x = x_1/z_1$ and $y = y_1/z_1$.

Let;

$$T = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{bmatrix} and\, p = (x, y, 1)$$

The location of the transformed keypoint p' is then computed with the following formula:

$$p' = Tp \tag{4.6}$$

where p'= (x′, y′, z′) is a 3D vector and its location on the projective plane is p' = (x′/z′, y′/z′). The x′, y′, and z′ are computed separately as follows:

$$\begin{aligned} x' &= x * m_0 + y * m_1 + m_2 \\ y' &= x * m_3 + y * m_4 + m_5 \\ z' &= x * m_6 + y * m_7 + m_8 \end{aligned} \tag{4.7}$$

Some of the keypoint detectors also estimate orientation for the keypoints. For such keypoints, the orientation angle should be calculated after the transformation to the deformed image .

Let θ is the orientation angle of the p. The orientation angle θ' of the p' is then calculated as follows:

$$\begin{aligned} a &= (x + cos(\theta)) * m_0 + (y + sin(\theta)) * m_1 + m_2 \\ b &= (x + cos(\theta)) * m_3 + (y + sin(\theta)) * m_4 + m_5 \\ c &= (x + cos(\theta)) * m_6 + (y + sin(\theta)) * m_7 + m_8 \\ \theta' &= \arctan(\frac{(\frac{b}{c} - y)}{(\frac{a}{c} - x)}) \end{aligned} \tag{4.8}$$

As it is stated in the introduction, the binary descriptor for the keypoint is computed from the patch surrounding it. Changes in texture of the patch lead to variations in binary descriptors. Each keypoint has an unique texture around it. So, the deformation on the image affects the texture of each keypoint differently. In order to observe these variations, we compute the binary descriptor for each keypoint k_i and its each transformation to synthetic images as illustrated in Figure 4.3.

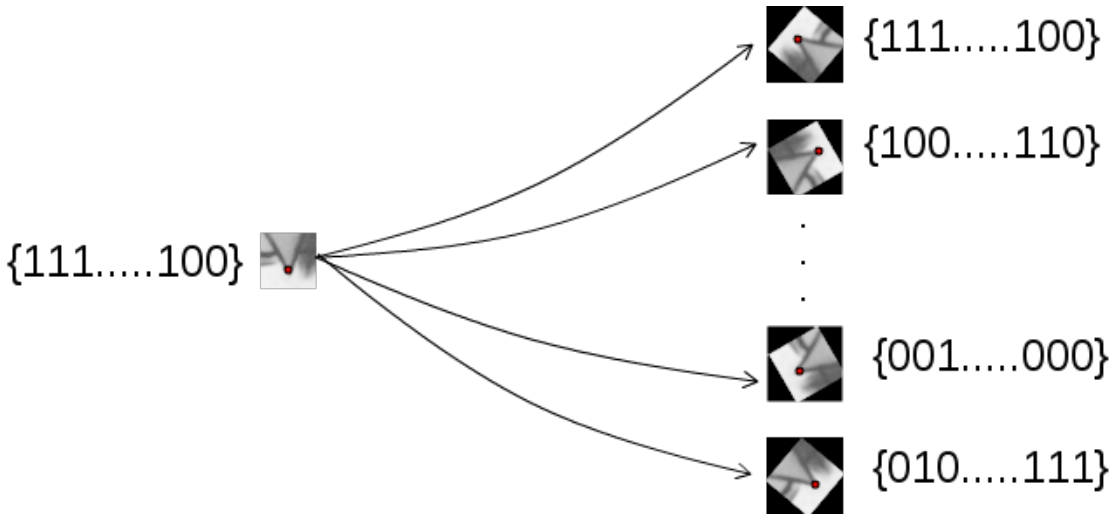


Figure 4.3. Each reference keypoint is warped to the synthetic images by ground-truth transformation matrices. The binary descriptor vector for the patch that is centred by the warped keypoint is then computed in each deformed image.

4.3. Experiments

To observe the keypoint stability, we measure the repeatability of the keypoint detector under various affine deformations. For testing binary descriptor stability, we compute the Hamming distance(introduced in Chapter 3) between descriptor of the each reference keypoint and descriptors of its deformations. We also observe bit variations for descriptor of the each reference keypoint per affine deformation.

4.3.1. Setup

Since only a single reference image is enough for obtaining different views of the planar scenes by using affine deformation model, we work with the first image of the *Graffiti* sequence from Oxford data sets by assuming that whole scenes are planar as in the study of [22, 23]. We convert it to grayscale which has size of 800x640. The detailed information about Oxford data sets is given in Section 5.3.1.

For synthetic image generation, three deformation parameters of four introduced in Section 4.2 are sampled which are the scale, in-plane rotation and tilt amount:

- Scale(λ) changes between 0.5 and 1.5 with steps of 0.1 ,
- In-plane rotation angle(ψ) changes between 0 and 359 degrees with steps of 5 degrees,
- Tilt amount(θ) changes between 0 and 80 degrees with steps of 5 degrees.

This sampling range is enough for generating different views of scenes that are close to the real so we do not need to sample tilt amount and also not need to sample other parameters more frequently. We use *warpPerspective()* method of the OpenCV library to generate synthetic images with randomly sampled deformation parameters in the interval given above. Three of the images which are synthetically generated from the reference Graffiti image by sampling different deformation parameters are shown in Figure 4.4.

(a) $\psi = 90°$ (b) $\theta = 60°$ (c) $\lambda = 0.5$

Figure 4.4. Three of the synthetic images generated from the reference Graffiti image. (a) is obtained by in-plane rotation of 90 degrees and its size is 640x800. (b) is generated by sampling tilt amount as 60 degrees and has a size of 400x640. (c) is sampled by half scaling of the reference image and has a half size of it.

As a keypoint detector, we use FAST since it is computationally efficient (Chapter 3). We detect approximately 1000 keypoints on each image. We compute BRIEF descrip-

tor of keypoints which performs well for matching together with FAST [7]. The OpenCV implementation is used for both FAST and BRIEF.

For quantifying descriptor variations under deformations, we compute the Hamming distance between descriptor of each reference keypoint and descriptors that are computed after it is transformed to each deformed image. The Hamming distance for the descriptor of each transformed keypoint gives the total number of unstable bits under the corresponding deformation. Moreover, in order to analyse variations in each bit of the reference descriptors under deformations, we create the binary matrix M_b for each reference keypoint. Each row r_M of matrix M_b represents a single descriptor bit individually and each column c_M belongs to one of the deformations in in-plane rotation angle. As column indexes increase from left-to-right in matrix M_b, in-plane rotation angle is incremented and ends up with 359 degrees. If the bit value changes under the corresponding deformation then we set entry (r_M,c_M) to 1, otherwise we set it 0. The number of 1s in a single column gives the Hamming distance to the in-plane rotation with degrees of the column index. For each bit, the number of 1s in the corresponding row shows the number of deformations that its value changes. Highly variant bits under deformations per keypoint can be found by calculating the fraction of 1s in row r_M and then sort the rows according to this measure like in the study of [4] which aims at to find discriminative binary tests . Each matrix M_b has 256 rows and 360 columns which is illustrated in Figure 4.5. For these experiments, in-plane rotation angle changes between 0 and 359 degrees with steps of 1 degree and we generate synthetic image for each sampled parameter.

4.3.2. Results

We show repeatability results of FAST keypoints for deformation parameters that we randomly sampled. We give results for in-plane rotation, scale and tilt amount in Figure 4.6, Figure 4.7 and Figure 4.8 respectively.

Repeatability through the in-plane rotation deformations is always greater than 70 percent which is shown in Figure 4.6. For each 90 degrees period, it shows similar behaviour. As mentioned in Chapter 3, FAST fits a grid on an image and compares each keypoint candidate with 16 pixels in its surrounding circle based on their intensities. Since the grid fits as the same as on the reference image in 90, 180, and 270 degrees, all reference keypoints repeat in such deformations which results in 100% repeatability.

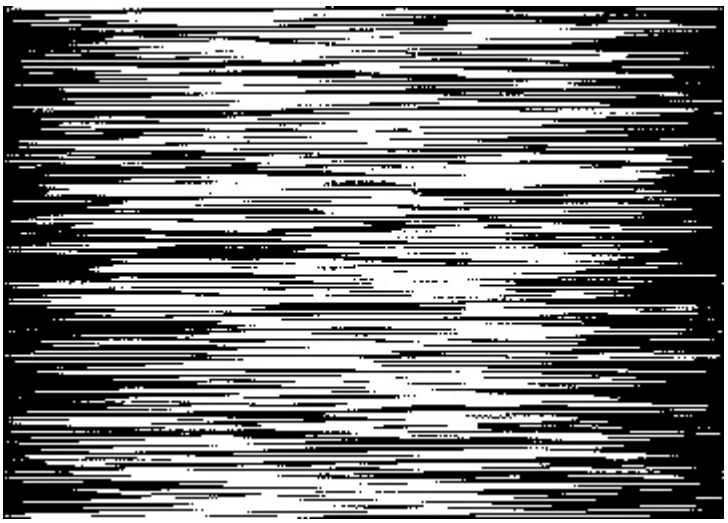


Figure 4.5. Visualization of binary matrix with height of 256 and width of 360 entries for one of the reference keypoints. Each row belongs to a single bit and columns represent in-plane rotation angle deformations for incrementing order in the interval of [0-359] degrees. Black entries represent stable bits and white ones belong to variant bits under corresponding deformation.

For scale variations depicted in Figure 4.7, repeatability decreases while amount of change in scale increases and it is always above about 40 percent. The repeatability shows very similar behaviour while scale becomes farther away from the reference in both ways. It can be seen that approximately half of the keypoints are repeated even the scale halved or be one-half. This shows that FAST is robust to variations in scale and has acceptable repeatability rates for different scales.

The repeatability is above 40 percent until the tilt amount is 60 degrees. It is inversely proportional with tilt amount and is very low from 60 degrees which is shown in Figure 4.8. It can be stated that FAST is less robust to changes in tilt amount than the in-plane rotation. However, if the variation in tilt amount is not too much, sufficient number of potential matches can be provided between different views of the scenes on the image.

We show the Hamming distance distribution of two reference keypoints k_i and k_j in Figure 4.9 under the same affine deformations only include in-plane rotation angle deformations between 0 and 359 degrees. It can be seen that for in-plane rotation around 180 degrees, keypoints k_i and k_j show different behaviours. For one of them, Hamming distances increase under such deformations while the other has just the opposite behaviour.

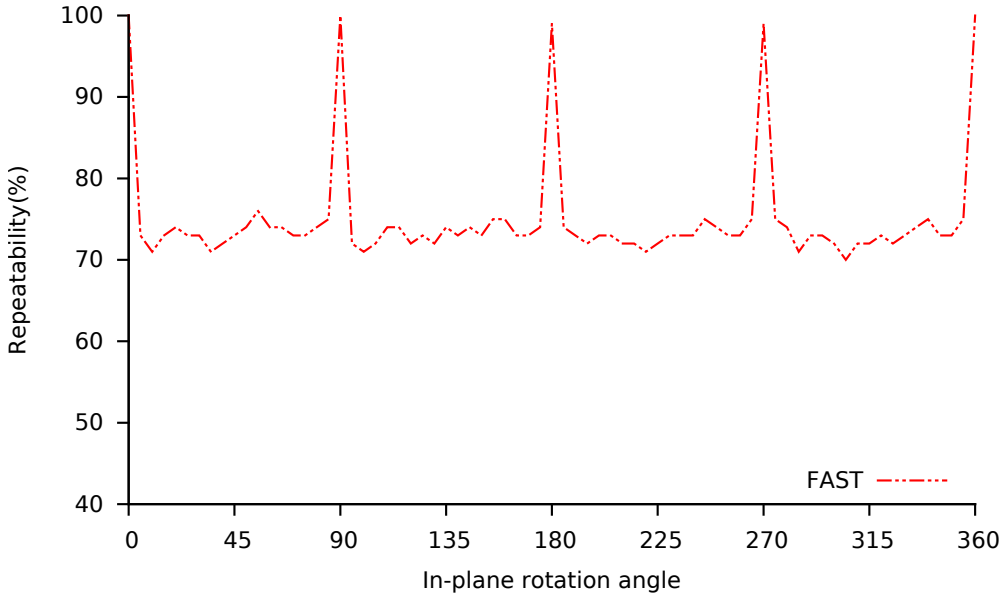


Figure 4.6. Repeatability of FAST keypoints for in-plane rotation angle variations.

Variations in first 10 bits of descriptors belong to keypoints k_i and k_j are shown by Figure 4.10 with their Hamming distance distributions. Although they have a similar Hamming distance distribution, changes in first 10 bits under the same deformations are quite different from each other.

4.4. Discussion

Despite the lack of orientation estimation and detection at multi-scales, the repeatability of FAST is acceptable under various affine deformations. More advanced keypoint detection methods such as AGAST [20] which estimates orientation of keypoints may have better repeatability results under the same deformations especially for in-plane rotations.

Although the same ground-truth transformation is applied, texture on the patch of each keypoint is affected differently since it is unique for each. This particularly leads to changes in the binary descriptor vector specifically to each keypoint which is computed from the corresponding patch. So, the number of unstable bits which is quantized by Hamming distance might be different under the same deformation for the keypoints as it is shown in Figure 4.9.

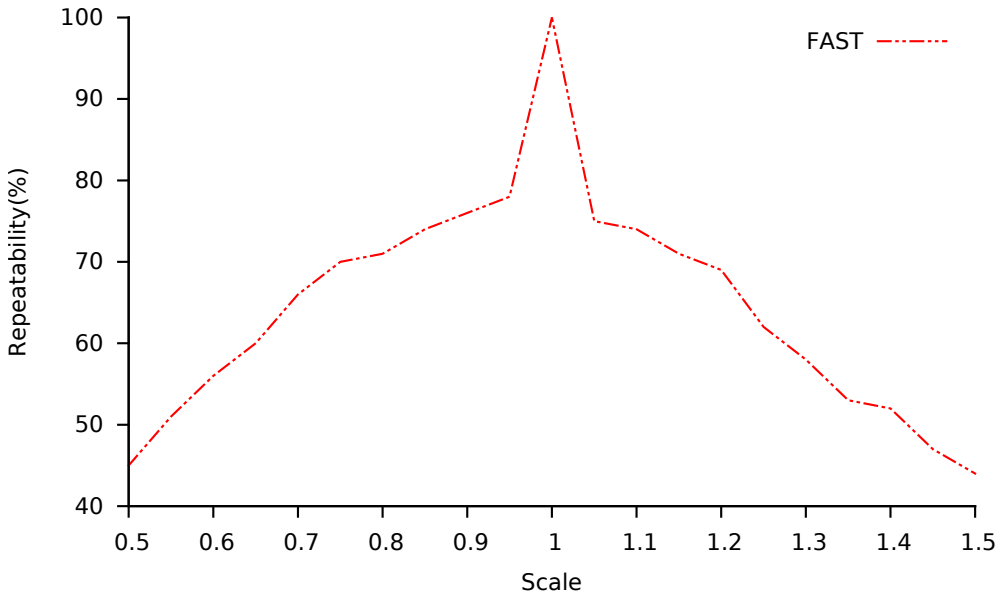


Figure 4.7. Repeatability of FAST keypoints for scale variations.

As it is stated in Chapter 3, binary tests which are the pair of pixels on the patch centred by the keypoint are used for binary descriptor computation by intensity comparisons on each test. Different tests are affected under the same deformation which leads to changes in different bits of the descriptor for each keypoint. Although more than one reference keypoints have similar Hamming distances to their own transformations, unstable bits of each are different which is also shown in Figure 4.10.

Due to variations in descriptors introduced in this chapter, the distance between descriptor vectors cannot always be sufficient to match keypoints correctly. Keypoints that belong to the same scene point in different images may have larger distance between themselves. Moreover, more than one keypoints may have the smallest distance to the single keypoint. These cases lead to incorrect matches between different views of the scenes on the image. In Chapter 5, we propose an approach by supplementing the distance information with a secondary and more distinctive score to obtain more correct matches.

4.5. Conclusion

Keypoint detection and matching are two fundamentals of the most computer vision applications. Generally, keypoint detectors have acceptable repeatability rates under

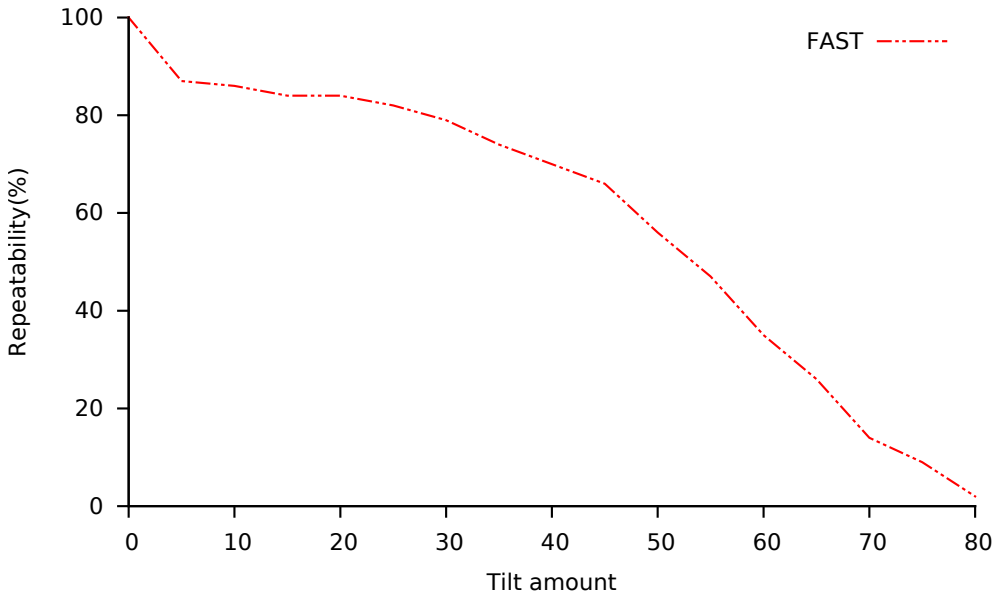


Figure 4.8. Repeatability of FAST keypoints for tilt amount variations.

the affine deformations. This shows that sufficient number of potential matches exists between keypoints of the same scene point in different images.

As discussed in the introduction, matching between keypoints is performed based on the distance between their binary descriptor vectors in real-time vision applications. Matching with binary descriptors has some difficulties due to their discrete nature. Larger changes in viewpoint and scale causes changes in descriptor bits. These variations are specific to each keypoint. This keypoint-based properties might be beneficial for keypoint matching besides the distance information.

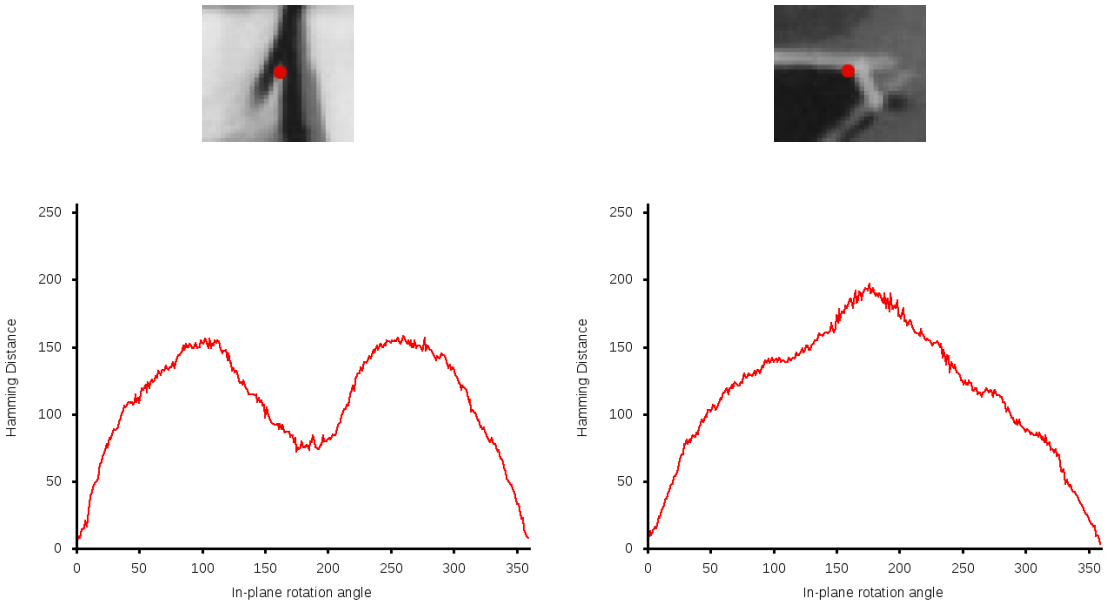


Figure 4.9. Hamming distance distributions of the two reference keypoints on Graffiti image along deformations by in-plane rotation angle variations. Texture around each keypoint is shown in the above of the corresponding Hamming distance distribution. Each reference keypoint k_i is transformed to synthetically generated images. BRIEF descriptors are computed for keypoint k_i and its transformations in deformed images. Hamming distance is then computed between descriptor of keypoint k_i and its each transformation. Around 180 degrees, they show quite different behaviours.

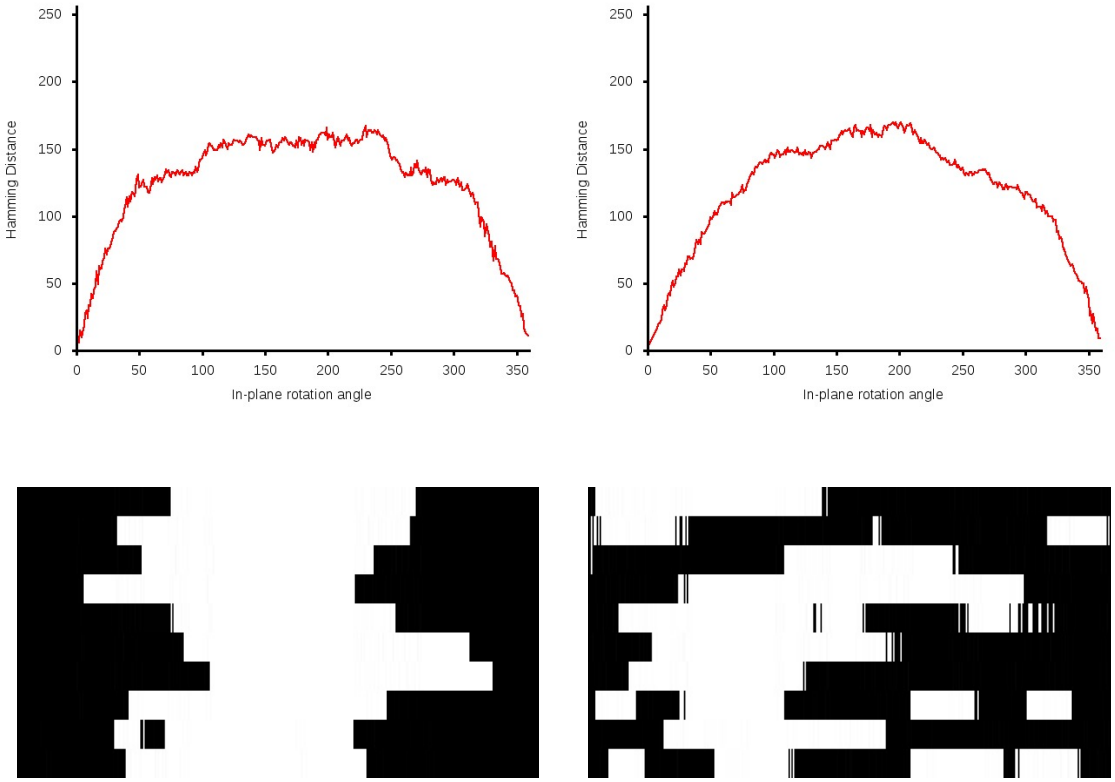


Figure 4.10. Variations in first 10 bits of two reference descriptors of keypoints on Graffiti image which show similar Hamming distance distributions along in-plane rotation angle deformations. First 10 rows of corresponding binary matrices(introduced in 4.3.1) are zoomed for visualization. Although they have very similar Hamming distance distribution, unstable descriptor bits are very different from each other under the same deformations.

CHAPTER 5

BINARY DESCRIPTOR MATCHING BEYOND THE NEAREST NEIGHBOUR

5.1. Introduction

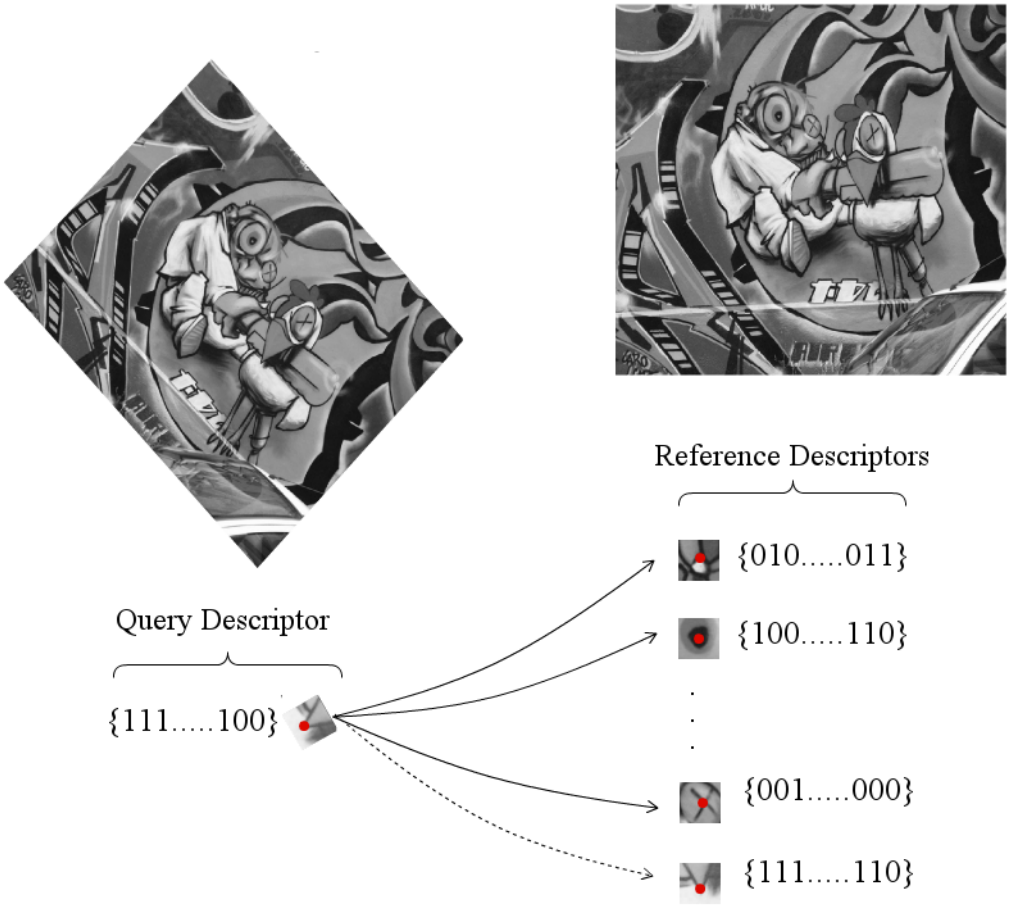


Figure 5.1. Keypoint matching with binary descriptors based on Hamming distance. For the descriptor of each query keypoint, the nearest neighbour is searched over the descriptors of reference keypoints according to Hamming distance. The reference keypoint that gives the smallest distance which is shown by dashed line is then assigned as match for the query keypoint.

Keypoint matching is establishing correspondences between the points in variously deformed images. Its aim is finding keypoints that belong to the same scene point on an image pair. Matching can be done based on keypoint locations if the geometrical relation between images is known. Since such relation cannot be obtained in real-time computer vision based systems, descriptors are used for keypoint matching. Matches are found according to distance between descriptor vectors of keypoints.

The binary descriptors (such as [7, 29, 16, 1, 32, 31, 17]) are extensively used in real-time applications for image matching and identification. Keypoint matching with binary descriptors is done by finding the reference descriptor that gives the smallest Hamming distance to the query descriptor. In other words, it is finding *the nearest neighbour* of the query descriptor over reference descriptors based on the Hamming distance as it is shown in Figure 5.1. As stated in Chapter 3, binary descriptors 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 neighbour within a set of reference descriptors.

Unfortunately, the binary descriptors are particularly sensitive to larger changes in viewpoint and scale. This is mainly due to their discrete nature. These variations are specific to each keypoint and the distance between descriptor vectors cannot always give correct matches as it is discussed in Chapter 4. For large binary descriptor data sets, more than one reference descriptors may be the nearest neighbour of the single query descriptor. Moreover, the distance for the correct match of the query descriptor might be a less more than the distance given by the nearest neighbour. However, it is relatively easy to create a list of the closest K near neighbours 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 neighbour as it is shown in Figure 1.1. Of course, the matches beyond the nearest neighbor are only reachable if we have a way of correctly reordering the match hypotheses to bring the correct one to the top. Therefore, we need to supplement the Hamming distance with a secondary and more distinctive match score to take advantage of these.

In designing this secondary match score, we make the following observations: Most binary descriptors, even the ones that optimize their representation, compute the same features for every keypoint. As shown recently by [4], while computationally appealing, this generic nature of the descriptors results in a weakness in discriminative power since the stability and the descriptiveness of features vary depending on the tex-

ture around a given keypoint. Globally optimizing the features for all keypoints is not as effective as picking unique features for each individual keypoint.

In this chapter, 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 neighbours according to the Hamming distance. In the second step, we rank these near neighbours according to a probabilistic and keypoint specific match quality score.

5.2. Approach

The usual approach for matching keypoints involves finding a single match hypothesis by locating the nearest neighbor in the descriptor space. Instead of this one-shot approach, we first locate possible match candidates that are near neighbors to the query in the descriptor space. We then evaluate each match candidate based on detailed statistics of the texture around the specific candidate keypoint. Figure 5.2 gives an overview of the proposed approach.

We learn the statistical model for each keypoint in an offline 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.

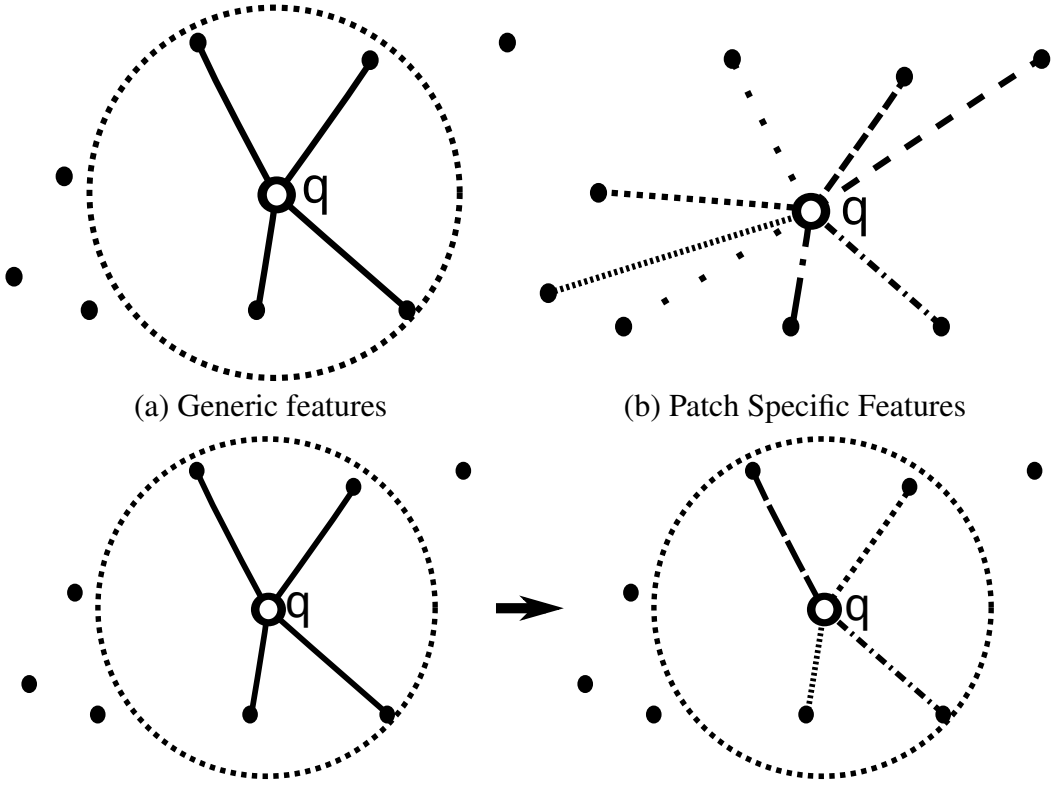
5.2.1. Modelling Descriptor Variations

Despite the robust nature of binary descriptors, the descriptor bits are not fully invariant to changes in perspective and lighting. As viewpoint and illumination changes, the affected bits depends on the texture around each keypoint. For a particular descriptor bit, 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 as it is shown in Figure 5.3.

Due to the complex nature of these interactions, variations in the descriptor value are more easily captured by a probabilistic conditional model such as

$$P(D \mid C = k_i) = P(d_1, d_2, \ldots, d_S \mid C = k_i)\,, \tag{5.1}$$

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



(a) Generic features (b) Patch Specific Features

(c) Proposed Approach

Figure 5.2. (a) Given a query descriptor q, if the descriptor uses the same features for each patch then it is possible to use an approximate nearest neighbour 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.

binary random variable representing the j^{th} 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 modelling the joint probability of the descriptor bits is not feasible. Following the representation proposed by [27], we split the descriptor into N groups of M bits such that $S = M \times N$ and assume independence between these:

$$P(D \mid C = k_i) = \prod_{j=1}^{N} P(D_j \mid C = k_i) \,, \tag{5.2}$$

where D_j represents the values of the bits in group j. This representation has $N \times 2^M$ pa-

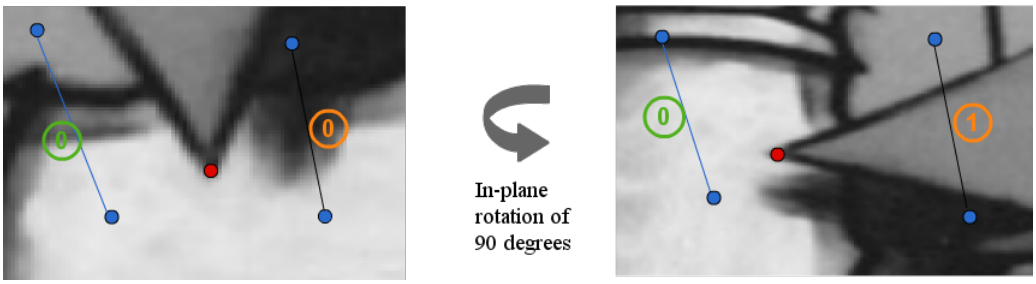


Figure 5.3. Two binary tests on the patches centred by the reference keypoint and its transformation under in-plane rotation of 90 degrees. Due to variations in texture on the pixels that belong to binary tests, the bit value given by one of the tests changes while the other remains stable.

rameters. We use two settings with $M = 4$ and $M = 8$. For 256 dimensional descriptors, these yield $N = 64$ and $N = 32$, requiring 1024 and 8192 parameters respectively.

The Naive Bayes modelling of keypoint appearance is not a novelty of our approach, but of [27]. Ours is the first to exploit it in a scalable way. To the best of our knowledge, there is no general theory that explains the efficiency of Naive Bayes, only empirical evidence. We note that, for classification problems, the exact modelling of probabilities is not required but only their ranks matter. As observed in [27], for keypoint recognition, Naive Bayes seems to outperform alternatives such as model averaging given enough training data and some regularization.

The descriptor bits can be assigned to groups in many ways. The simplest possibility is to form the bit groups by assigning the consecutive descriptor bits to the same group. For descriptors like BRIEF [7] that exploit randomness in bit selection, this is a natural choice as there is no reason to favor one grouping scheme over another. For descriptors like BRISK [16] with more structured features, 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. Figure 5.4 shows bit grouping for $M = 4$ and $N = 64$.

For each keypoint, we generate synthetic training samples using the affine deformation model (explained in detail in Section 4.2) in order to observe bit variations. We sampled all parameters of the model as follows:

- Scale varies between $\frac{1}{\sqrt{2}}$ and $\sqrt{2}$.
- In-plane rotation angle varies between -30 and $+30$ degrees.

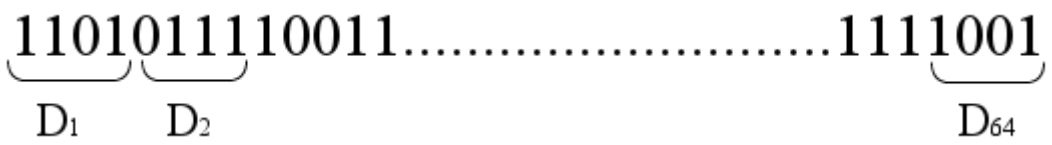


Figure 5.4. Splitting 256 dimensional descriptor into 64 groups of 4 bits. Each consecutive 4 bits form a group. Bit groups are independent from each other which results in 1024 parameters require per keypoint.

- Tilt amount varies between 0 and 60 degrees.
- Tilt angle varies between 0 and 180 degrees.

We generate roughly 200000 synthetic training images, yielding the same number of training descriptors for each keypoint. For each bit group, the observation probabilities of Equation 5.2 are inferred from the training set 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 for each group. We just count the number of times each outcome is observed within the training set and normalize by the total number of training samples as it is illustrated in Figure 5.5. To counter the adverse effect of zeros in the estimated probabilities, we start counting from one as suggested by [27], which is equivalent to Laplace smoothing with the smoothing parameter set to 1.

5.2.2. Keypoint Specific Scoring of Match Hypotheses

For each keypoint to be matched, we first obtain the list of K near neighbors based on the Hamming distance to the query descriptor. This short list of K descriptors 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

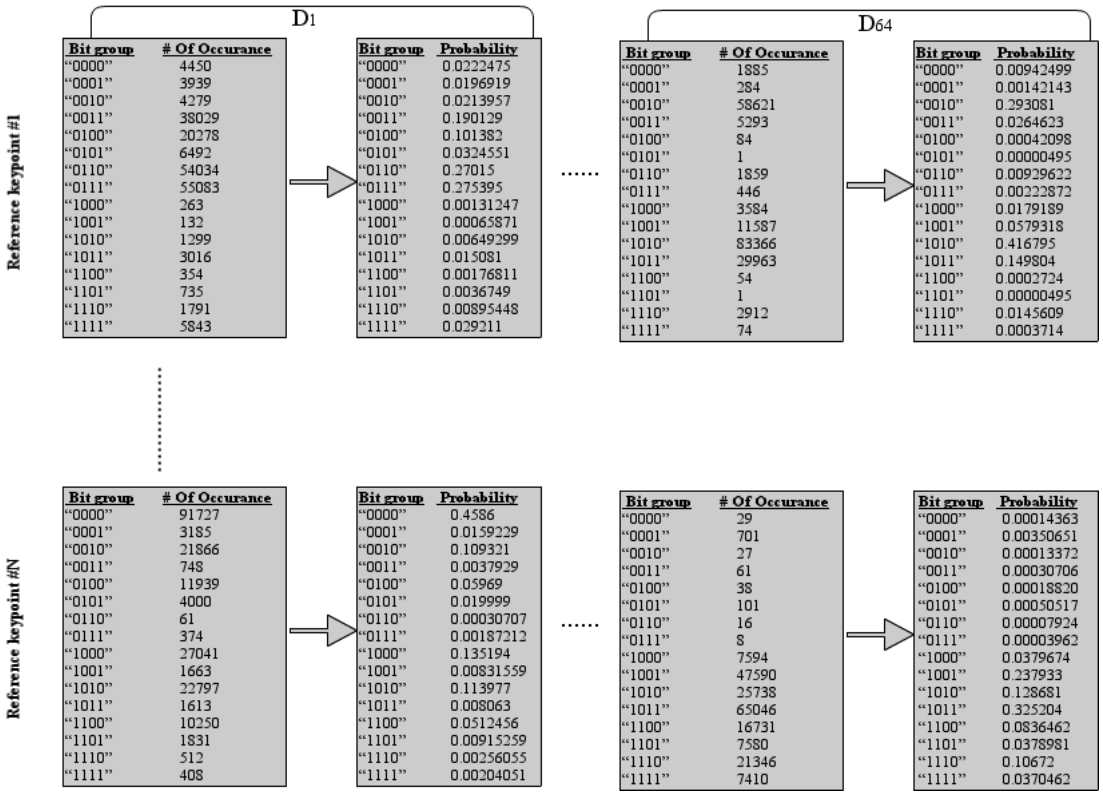


Figure 5.5. Obtaining observation probabilities for each reference keypoint k_i in training phase. 256 dimensional descriptors are split into groups of 4 bits which results in 64 different groups per keypoint k_i. For each bit group of keypoint k_i, there are 16 possible outcomes. The number of occurrence is counted for each possible outcome during offline training. Finally, each of them is normalized by the number of training samples to get observation probabilities.

observing the query descriptor according to Equation 5.2:

$$\text{score}(Q, k_i) = -\left|Q - D^i\right|_H + \log \prod_{j=1}^{N} P\left(Q_j \mid C = k_i\right) \ , \tag{5.3}$$

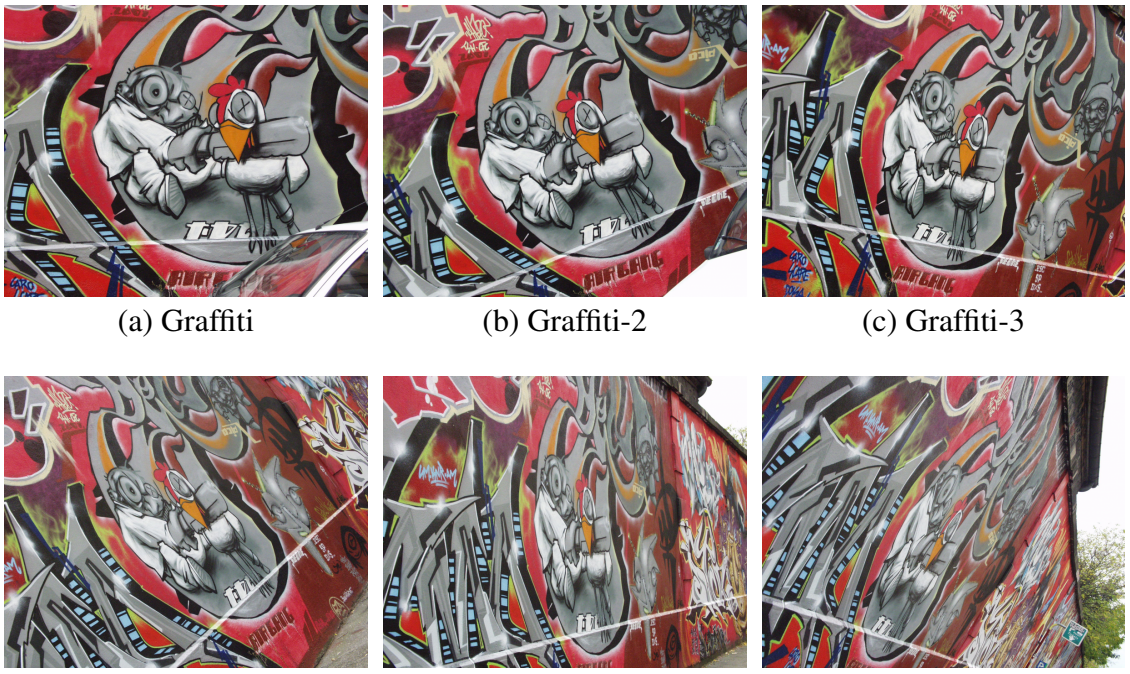
where Q is the query descriptor, D^i is the descriptor for the reference keypoint i, and $|X - Y|_H$ is the Hamming distance between X and Y.

5.3. Experiments

To test the ability of our approach in recovering the correct matches from the near neighbour list, we have performed five sets of experiments. The first one is to measure the effect of the near neighbour list length K on the recognition performance. The second one measures the recognition rate performance with different weighted components of keypoint specific and probabilistic score(Equation 5.3). The third one measures the recognition rate over ground truth correspondences. The fourth 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 large number of reference images and an approximate nearest neighbour algorithm is used to compute the near neighbour list.

Figure 5.6. Reference images in the Oxford dataset.

5.3.1. Setup



(a) Graffiti (b) Graffiti-2 (c) Graffiti-3

(d) Graffiti-4 (e) Graffiti-5 (f) Graffiti-6

Figure 5.7. Graffiti image sequence in the Oxford dataset. (a) is the reference image and the others are test images. The geometrical relation between the reference image and test images are known. From (b) to (f), the deformation on the reference image increases.

In all experiments we worked with Oxford datasets by converting images to gray scale. Oxford dataset has 8 different image sequences and the reference image of each sequence is shown in Figure 5.6. Each image sequence contains the reference image and five test images. Ground-truth homography is also provided between the reference image and each test image. Test images are labelled from 2 to 6. As the identifier of the test image increments, the level of the deformation also increases as it is shown for the Graffiti sequence in Figure 5.7.

5.3.1.1. Effect of the Near Neighbour List Length

To demonstrate the effect of the near neighbor list length K on the recognition performance, we perform an experiment on the *Graffiti* and *Wall* sequences of the Oxford datasets. We detect approximately 1000 keypoints on the reference image of the sequence and transform their coordinates to the rest of the images using the provided 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 Equation 5.3. We then 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 groups size M set to 8.

5.3.1.2. Effect of the Weighted Keypoint Specific Scoring

As it is explained in Section 5.2.2, our keypoint specific and probabilistic score(see Equation 5.3) is composed of two components. In order to observe the effect of the different weights of these components in score computation, we perform recognition rate experiments on Graffiti and Wall image sequences for different weights with BRIEF correspondences by using top 10 near neighbours with 8 bit grouping. The weighted score is shown by the following equation:

$$\text{score}(Q, k_i) = (1-\alpha) * (-\left|Q - D^i\right|_H) + \alpha * (\log \prod_{j=1}^{N} P\left(Q_j \mid C = k_i\right)), \qquad (5.4)$$

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

5.3.1.3. Recognition Rate over Ground Truth Correspondences

We measure the recognition rate over ground truth correspondences for four image sequences which are Boat, Bikes, Graffiti, and Wall. We preferred these sequences since different type of deformations are applied for each as follows:

- For the Bikes sequence, test images are blurred.

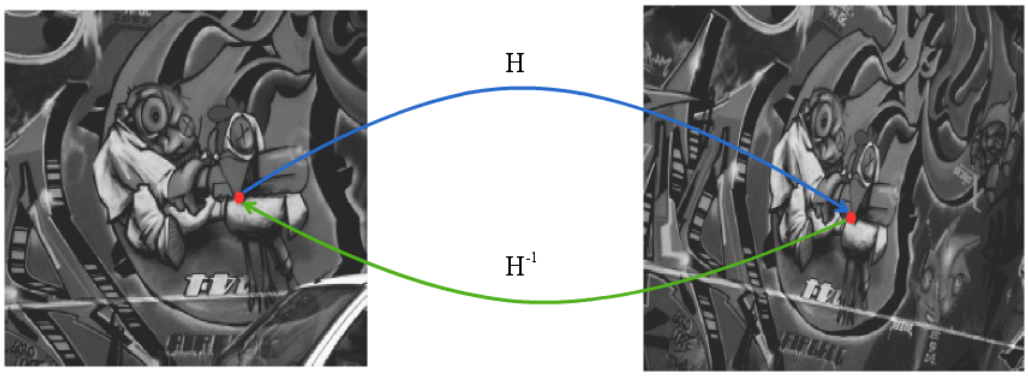
- In-plane rotation angle and scale varies for the test images of the Boat sequence.
- In-plane rotation angle, scale and tilt parameters are sampled for the test images of Graffiti sequence.
- For the Wall sequence, only tilt parameter is sampled for the test images.

We perform recognition rate experiments for five different binary descriptors which are BRIEF [7], BRISK [16], FREAK [1], LATCH [17], and ORB [29]. BRIEF, ORB, and LATCH are 256 dimensional descriptors, while others are 512-bit long. We set near neighbour list length $K \in \{5, 10, 20\}$ but we concentrate on the results with the $K = 10$ according to result of the previous experiment which is shown in Section 5.3.2 and we used three values for the bit group size ($M \in \{4, 6, 8\}$). We obtain correspondences by the same procedure that we apply for measuring the effect of the near neighbour list length in Section 5.3.1.1.

5.3.1.4. Improving the Inlier Ratio Characteristics

The previous experiments only 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. To obtain correspondences from test image to reference image, we take the inverse of the ground truth homography and transform each keypoint in the test image to the reference image by inverse homography as it is illustrated in Figure 5.8. For each transformed test keypoint, the Euclidean distance is then computed with all reference keypoints. The reference keypoint is assigned as potential match of the test keypoint if its distance is the smallest and less than 2 pixels. For some test keypoints, any of the reference keypoints might not be potential match. After we determine potential correspondences, we rank them by their negative descriptor distance and measure the ratio of the correct matches.

This inlier ratio changes as more correspondences are included from the ranked list. Since some test keypoints have no potential match in the reference image, they can never be correctly matched. Ideally, these should have the largest descriptor distances and end up at the bottom of this list.



(a) Graffiti (b) Graffiti-3

Figure 5.8. Obtaining correspondences from test image to reference image. We warp keypoints on the test image to the reference image by taking the inverse of the ground-truth homography. For each transformed test keypoint, the Euclidean distance is computed with all keypoints on the reference image. The reference keypoint which gives the distance that is the smallest and less than 2 pixels is is assigned as potential match.

The inlier ratio curve obtained as above is what matters for algorithms such as PROSAC[9] that estimates a geometric model for 3D registration or augmented reality applications. To show the benefits of our approach, we perform three sets of measurements. First, as a baseline, we rank the nearest neighbor matches using the descriptor Hamming distance and record the inlier ratios. Then we measure the inlier ratios using only the nearest neighbour match but rank the matches according to the keypoint specific score obtained from the Equation 5.3. This shows the effect of keypoint specific scoring without increasing the final inlier count. Finally, we both pick the best match in the ten near neighbor list and rank these correspondences according to scores of Equation 5.3. We perform experiments on the test image three of each sequence, since it is challenging enough without being too hard. The same binary descriptors are used with experiments for the recognition rate over ground-truth correspondences explained in Section 5.3.1.3.

5.3.1.5. Keypoint Matching with Locality Sensitive Hashing

For some vision tasks, such as augmented reality on multiple pages of a magazine, 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 for such a case, we concatenate the descriptors from all reference images from *Graffiti*, *Boat*, *Bikes*, *Wall*, plus the four remaining images in the Oxford datasets. The total number of reference descriptors is around 8000. As in the experiments of Section 5.3.1.4, we compute the inlier ratio curves on the third test image but this time we compute the list of ten near neighbors with Locality Senstive Hashing (LSH) instead of brute-force search. We use the FLANN [25] implementation of LSH that is available in OpenCV with 12 tables, a key length of 20, and a multi-probe level of 2.

5.3.2. Results

Effect of near neighbour list length to the recognition rate is shown in Figure 5.9. For the test image 2 and 3 of *Wall* sequence, extending the near neighbour list length K does not effect the recognition rate which is already close to 100%. For the other test images, expanding the near neighbour list provides to have more correct matches. After $K = 10$, it can be seen that there is no significant improvement for the recognition rate for these images. For the test image 4, the recognition rate is increased from 75% to 83% with $K = 10$ and remains stable while K extends. As it can be observed, enlarging K yields improved recognition rates especially for the *Graffiti*. We have significant improvement for the recognition rate with $K = 10$ and there is no important increase with K more than 10 like *Wall* . With $K = 10$, recognition rate for the test image 2,3, and 4 is improved to 76%, 45%, and 5% respectively which is more than two times of the $K = 1$ performance. So, since the improvement is less pronounced after $K = 10$, we set K to 10 for the remaining experiments. We also perform and show all experiments by setting K to 5 and 20, but we concentrate on the results obtained with $K = 10$.

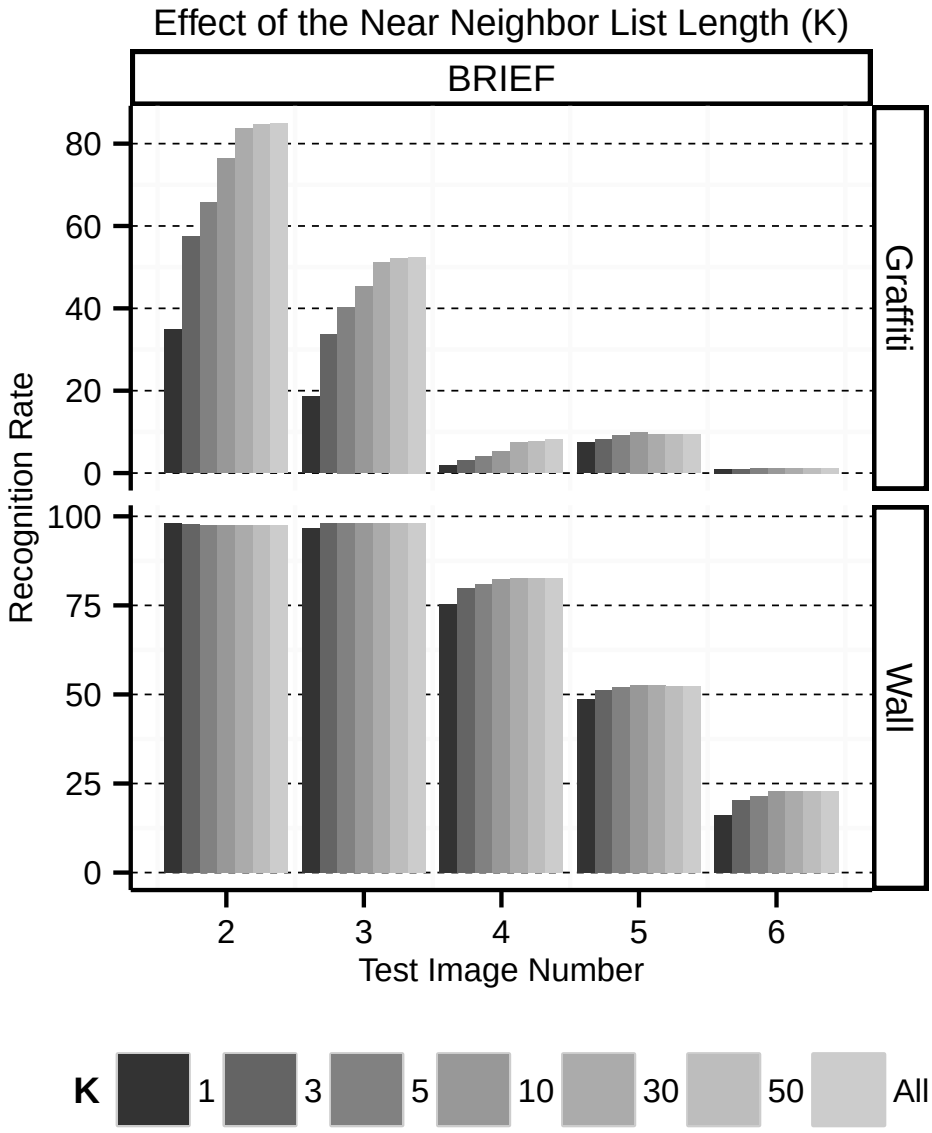


Figure 5.9. The recognition rates for the BRIEF (without orientation estimation) descriptor increase as we consider more near neighbors. This is especially true for *Graffiti*, where the baseline performance ($K = 1$) is low and the test images exhibit both scale and rotation changes. The difference between considering the first ten near neighbors or all reference keypoints is relatively low. At $K = 10$, the number of correct BRIEF matches for Graffiti–3 increases to 385. This is still less than the maximum potential number 517 (See Figure 1.1), but substantially better than 159, the baseline number achieved by just the nearest neighbor.

Effect of the different weights of components in keypoint specific score to the recognition rate is shown in Figure 5.10. α around 0.5 gives the best recognition rate results in most of the cases. So, we set α = 0.5 that is two components in keypoint specific score have equal weight in computation.

Figure 5.11 - 5.14 show the recognition rate results for four image sequences (*Bikes, Boat, Graffiti, Wall*) and five descriptor types (*BRIEF, BRISK, FREAK, LATCH, ORB*) by using $K = 10$ and the bit group size of 4, 6 and 8. We also show recognition rate results for using $K = 5$ in Appendix A between Figure A.1 - A.4 and $K = 20$ in Appendix A between Figure A.5 - A.8. For each bin of histogram, the leftmost column shows the result for the nearest neighbour(baseline), the second, the third, and the rightmost column represents the recognition rates obtained by using Equation 5.3 with bit group size of 4,6, and 8 respectively.

For 1000 reference and 928 query keypoints, the total brute-force search time for the nearest neighbour is 4.0 milliseconds using the POPCNT [1] instruction on a 64-bit laptop CPU. At $K = 10$, brute-force search with our approach takes 4.2 milliseconds irrespective of the value of size of the bit groups.

As it is shown in Figure 5.11, for the *Bikes* sequence, our recognition rate results are very close to the nearest neighbour results for BRIEF, LATCH, and ORB correspondences. Except the test image 6, almost all correspondences are correct matches. For BRISK and FREAK correspondences, we increase the recognition rate for all test images and generally 8-bit length grouping gives the best results. For the test image 4 with FREAK correspondences, we improved the recognition rate from 77% to 84% by our approach using groups of 8 bits.

For the *Boat* sequence, it can be seen from the Figure 5.12 there is no improvement after the test image 2 with BRIEF correspondences. For the other descriptors, the results are improved by a large amount for the test image 2 and 3 except for the second test image with ORB correspondences. For example, we increase the recognition rate from 34% to 61% for the test image 3 with LATCH correspondences by bit grouping of 6 or 8 bits. However, even the ten near neighbour list does not include any correct matches for the test image 5 and 6 with any of the descriptors.

[1] The POPCNT instruction computes the number of bits that are set to 1.

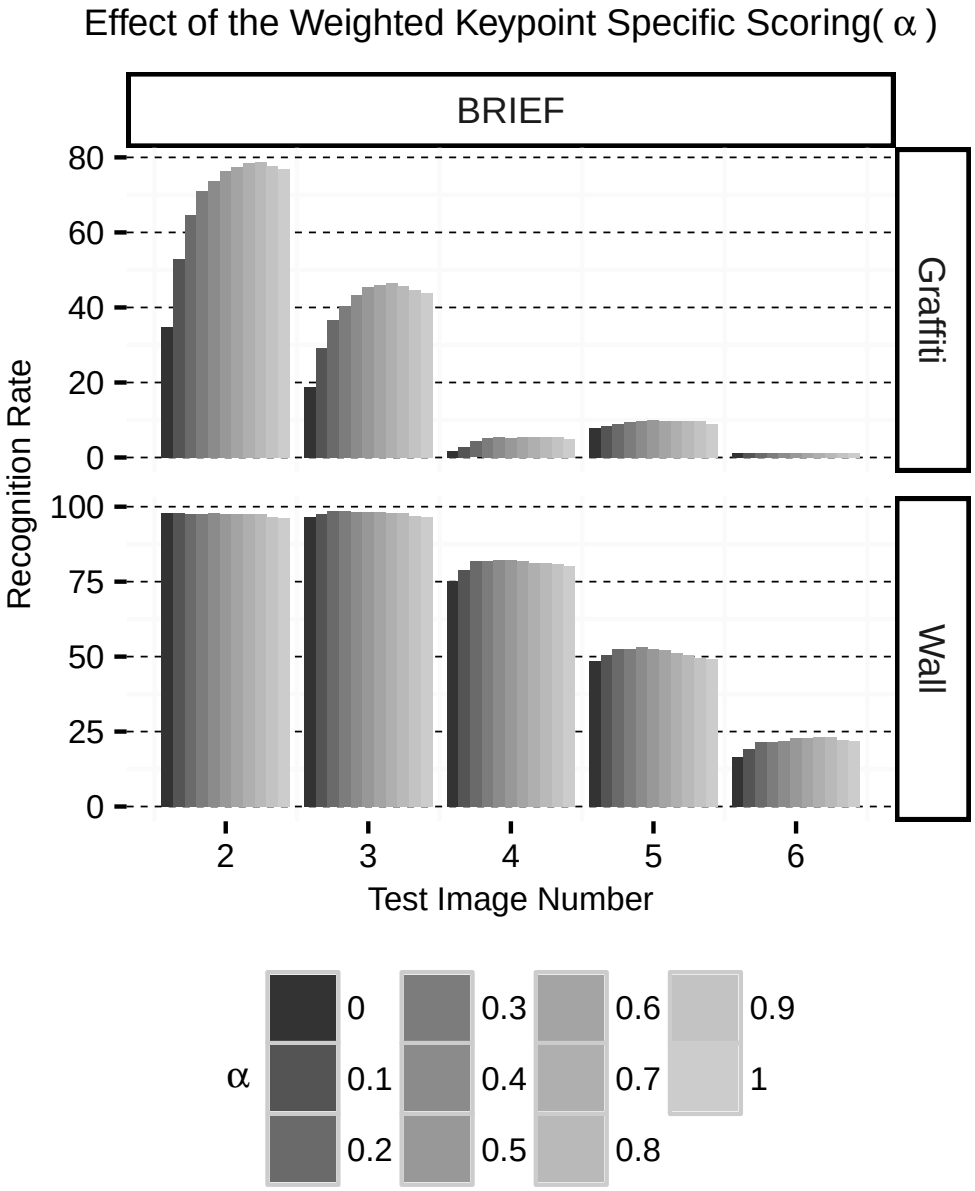


Figure 5.10. The effect of the weighted keypoint specific scoring on recognition rate. The recognition rate is obtained by using only the nearest neighbour when $\alpha = 0$. For the other weights, recognition rate is measured by using the top 10 near neighbours and reranking these near neighbours by weighted keypoint specific score(Equation 5.4). In most cases, weights of two components that are equal or close to each other give better recognition rate results than the ones are far away from each other. Generally, α around 0.5 results in higher recognition rates than the other weights.

We increase the recognition rate for the *Graffiti* sequence with all descriptors which is depicted in Figure 5.13. Especially for test image 2,3, and 4 we have significant improvement. With BRIEF correspondences, we at least doubled the recognition rate for these images by using 8 bits per group. For example, recognition rate of test image 3 is increased from 18% to 45%. For the test image 5 and 6, it can be seen that the number of correct matches is very low even with the ten near neighbours but we generally have better results than the baseline performance with any of the descriptors.

As it can be seen from the Figure 5.14, for the test image 2 of the *Wall* sequence, recognition rate performance of the baseline and our approach are close to each other and nearly 100% with BRIEF, LATCH, and ORB correspondences. For the other test images, we have an important improvement especially for 8-bit grouping with all descriptors. For example, the recognition rate is increased from 52% to 66% for the test image 4 with LATCH descriptors.

Inlier ratio values for the third test image with a single reference image and brute-force matches are shown for four images and five descriptors in Table 5.2 and graphs are depicted separately in Appendix B between Figure B.1 - B.16. We perform two sets of inlier ratio experiments as we explained in Section 5.3.1. We measure the inlier ratio by ranking the nearest neighbour matches by the keypoint specific score of Equation 5.3 with bit groups of 4, 6, and 8 (named as *Reweighted NN*). Moreover, we measure the inlier ratio by ranking the top $K = \{5, 10, 20\}$ near neighbour matches by the keypoint specific score of Equation 5.3 with bit groups of 4, 6, and 8 (named as K *NNs*). For both experiments, we make comparison with the inlier ratio that is measured by ranking the nearest neighbour matches only by the descriptor distance. Table 5.1 gives figure numbers in Appendix B corresponding to different inlier ratio experiments with various image sequences.

Table 5.1. Figure numbers belong to the inlier ratio experiments for the third test with a single reference image and brute-force matches.

Experiment / Image Sequence	*Boat*	*Bikes*	*Graffiti*	*Wall*
Reweighted NN	Figure B.1	Figure B.2	Figure B.3	Figure B.4
Five NNs	Figure B.5	Figure B.6	Figure B.7	Figure B.8
Ten NNs	Figure B.9	Figure B.10	Figure B.11	Figure B.12
Twenty NNs	Figure B.13	Figure B.14	Figure B.15	Figure B.16

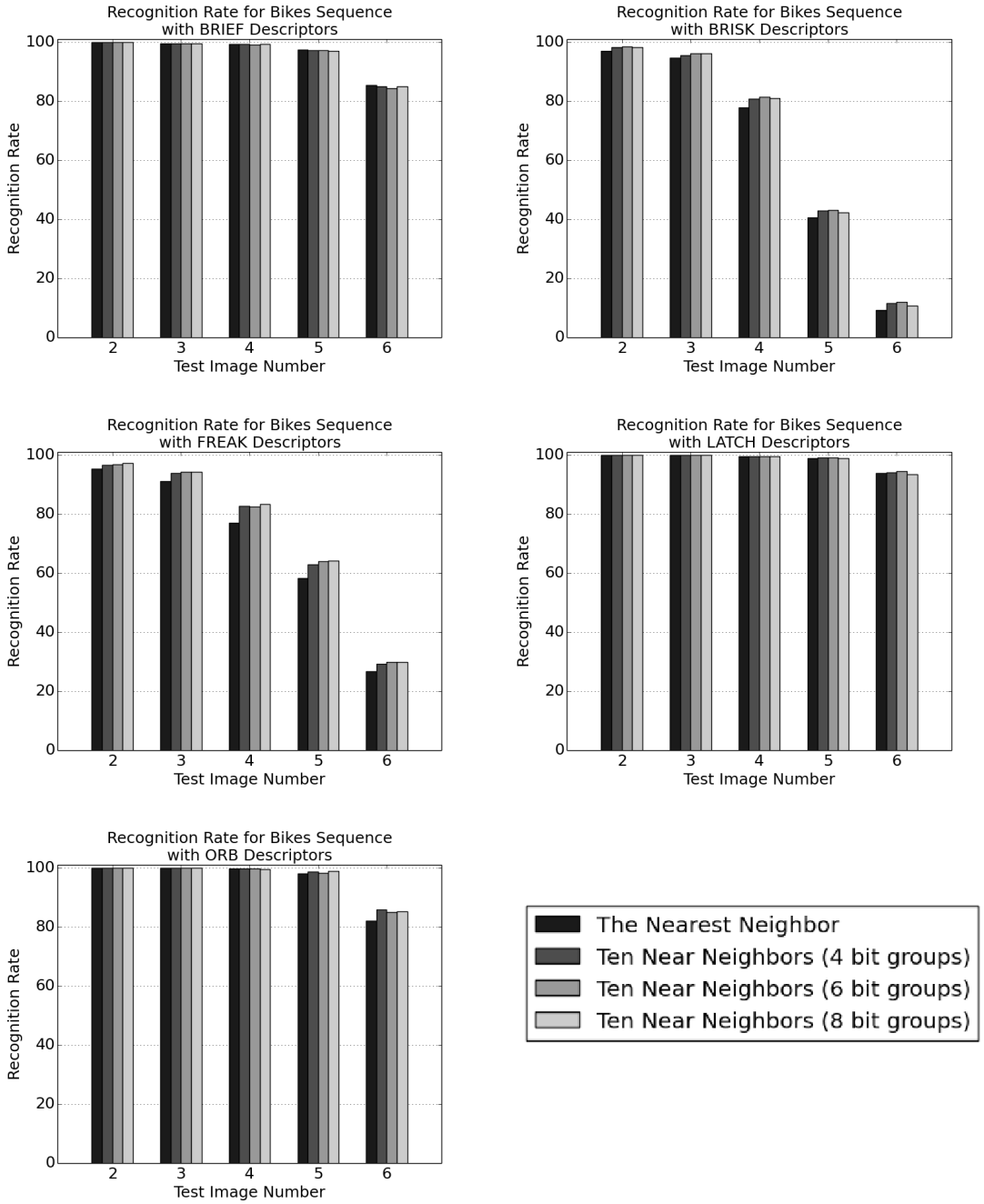


Figure 5.11. Recognition rates for the Bikes sequence with various descriptors obtained by using just the nearest neighbour and by ranking the ten near neighbours with keypoint specific scoring (Equation 5.3).

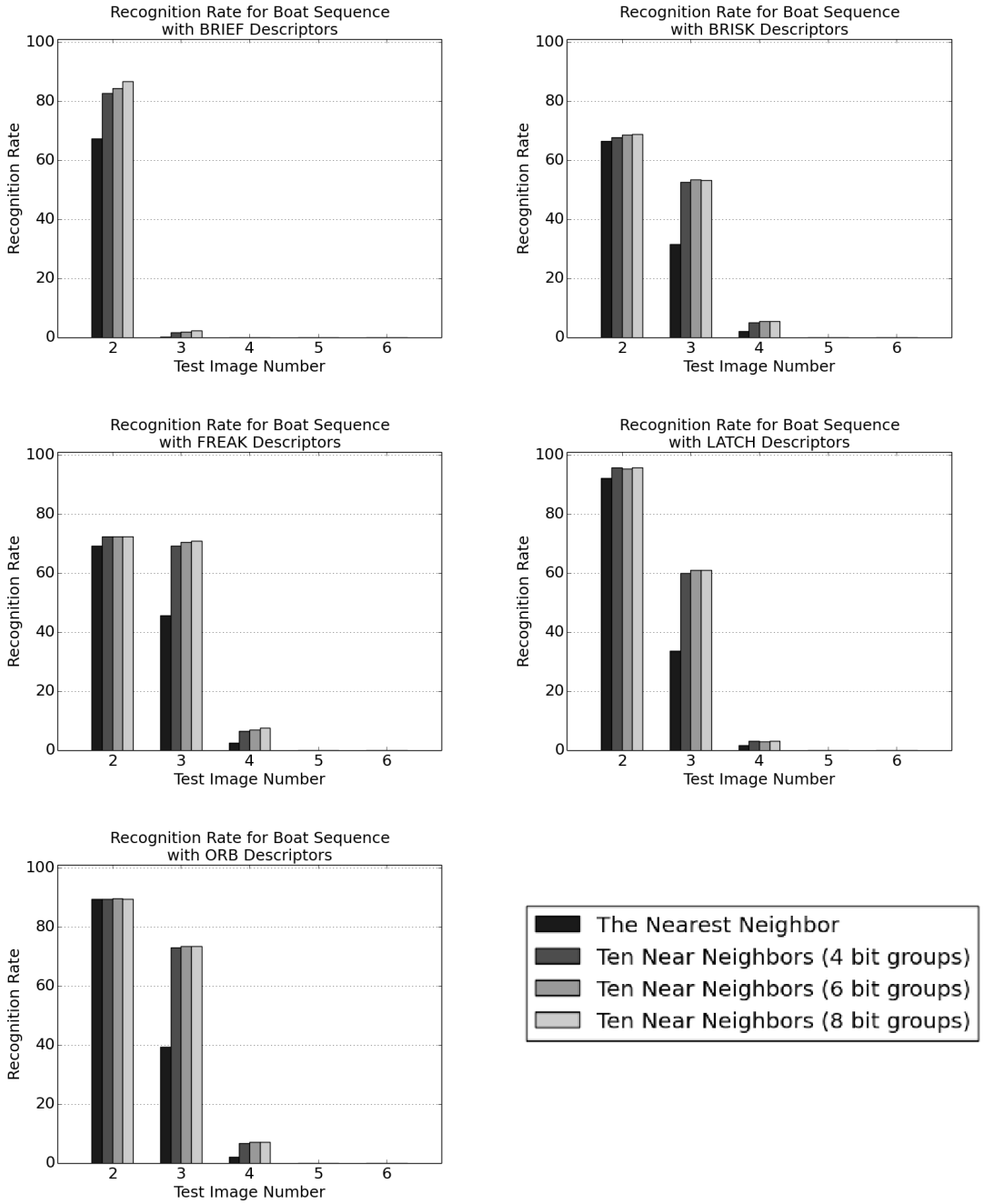


Figure 5.12. Recognition rates for the Boat sequence with various descriptors obtained by using just the nearest neighbour and by ranking the ten near neighbours with keypoint specific scoring (Equation 5.3).

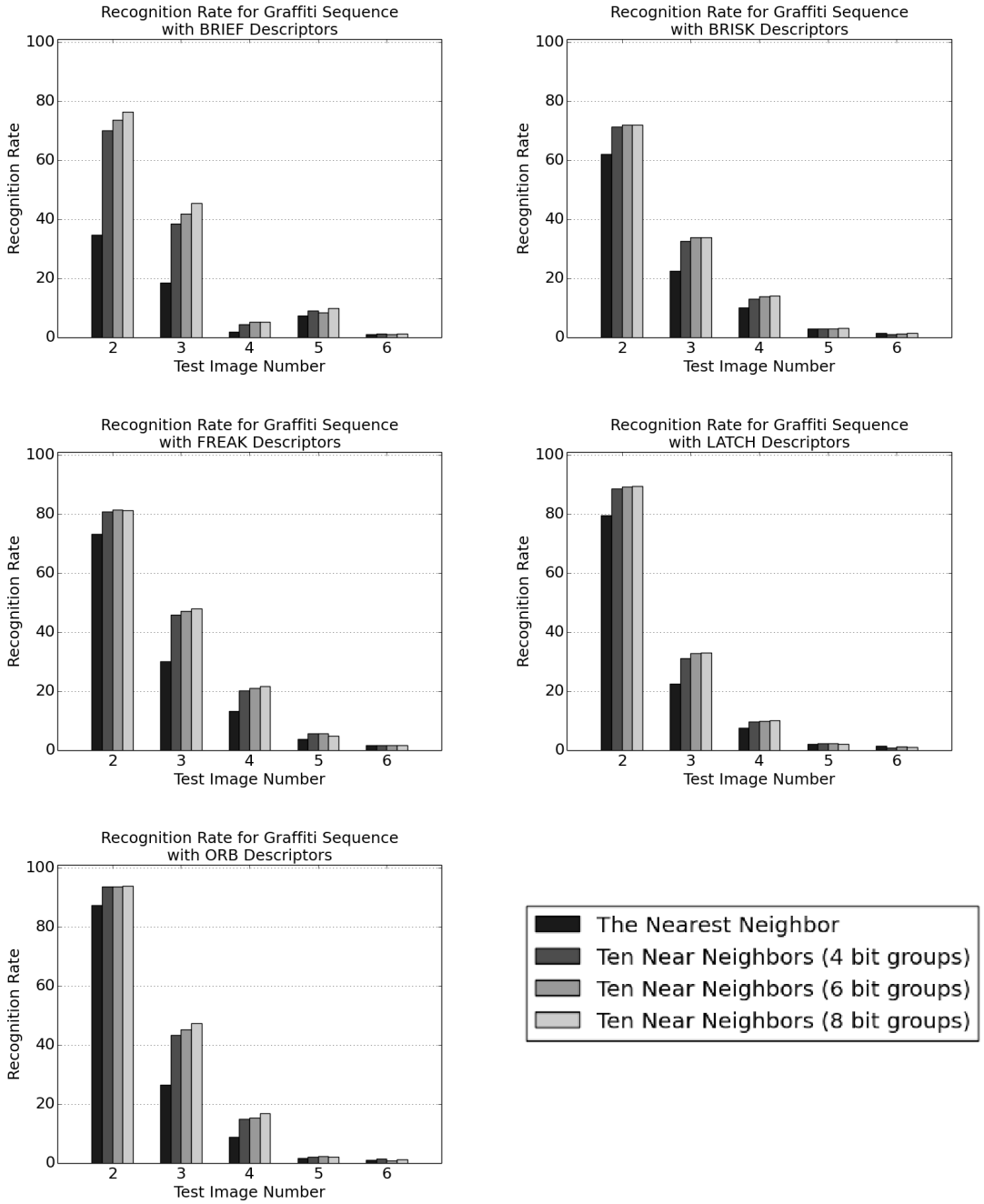


Figure 5.13. Recognition rates for the Graffiti sequence with various descriptors obtained by using just the nearest neighbour and by ranking the ten near neighbours with keypoint specific scoring (Equation 5.3).

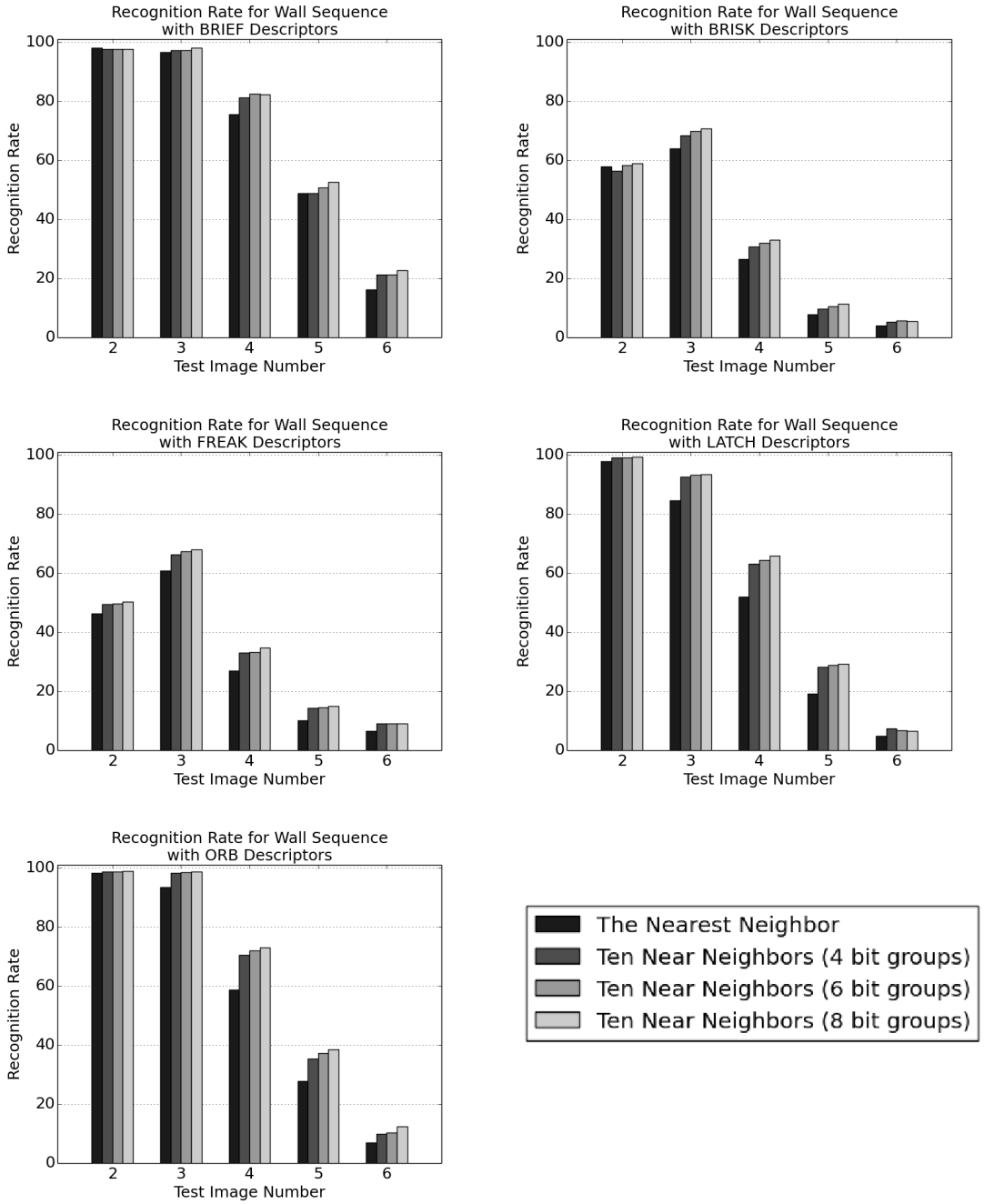


Figure 5.14. Recognition rates for the Wall sequence with various descriptors obtained by using just the nearest neighbour and by ranking the ten near neighbours with keypoint specific scoring (Equation 5.3).

In each graph, the green curve shows the inlier ratio obtained by ranking the nearest neighbour matches only by the descriptor distance(baseline). The yellow, the red, and the blue curves represent the inlier ratio obtained by using keypoint specific score computed from the Equation 5.3 with bit grouping of 4, 6, and 8 bits respectively.

Except with BRIEF correspondences, we increase the total number of inliers and obtain higher inlier ratio over all correspondences for the *Boat* as it is shown in Figure B.10. For example, the overall inlier ratio with FREAK correspondences is increased from 22% to 35% and 37% by groups of 4 and 8 bits respectively. Also the reweighted nearest neighbour curves have the improvement brought only by using the scores of Equation 5.3. As it can be seen from the Figure B.2, we improve the inlier ratio to 70% and 72% from 43% for the best 300 FREAK correspondences with grouping of 4 and 6 bits respectively.

For *Graffiti*, the baseline inlier ratio among the best 250 BRIEF correspondences is 21%. This rate is increased to 32%, 33%, and 34% by our approach with bit grouping of 4, 6, and 8 bits respectively which can be seen from the Figure B.3. With other descriptors, inlier ratio for the best few hundred correspondences and overall inlier ratio are also improved like BRIEF. As it is illustrated in Figure B.11, the overall inlier ratio for the BRISK correspondences is increased from 14% to 22% with 4-bit grouping.

We have significant improvement in inlier ratio for less number of top-ranked correspondences and overall inlier ratio with BRISK and FREAK correspondences for the *Bikes*. For example, we increase the inlier ratio among the best 300 BRISK correspondences from 57% to 67% which can be seen from the Figure B.1 with grouping of 6 bits. Moreover, the overall inlier ratio with FREAK correspondences is increased from 30% to 39% by our approach with grouping of 8 bits as it is illustrated in Figure B.9. For the other descriptors, we also have improvement for inlier ratios but not too much.

As it is shown in Figure B.12, we have approximately 5% increase in inlier ratios for the *Wall* with BRISK, FREAK, LATCH, and ORB correspondences. For example, we increase the overall inlier ratio with ORB correspondences from 33% to 37% with 8-bit length groups. With BRIEF correspondences our performance is slightly better than the baseline performance in which inlier ratios are very close to the each other.

Table 5.2. Inlier ratio values when matching the third test image to the reference image in each image sequence. Results are obtained by ranking the nearest neighbor matches only by the descriptor distance (NN), by the keypoint specific score of Equation 5.3 (Reweighted NN $\rightarrow$ RNN), or ranking the K near neighbor list matches by Equation 5.3 (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.

	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**

Inlier ratio values for the third test image with eight reference images are shown in Table 5.4 and graphs are depicted separately in Appendix C between Figure C.1 - C.16. In these experiments, matches are obtained by LSH instead of in a brute-force way as in the previous experiments. With parameters given in the Section 5.3.1.5, the LSH precision is around 90% in these experiments. In other words, the nearest neighbour found by LSH is the same as found by the brute force nine out of ten times. Table 5.3 gives figure numbers for different inlier ratio experiments with various image sequences.

Table 5.3. Figure numbers belong to the inlier ratio experiments for the third test image with eight reference images and LSH matches.

Experiment / Image Sequence	*Boat*	*Bikes*	*Graffiti*	*Wall*
Reweighted NN	Figure C.1	Figure C.2	Figure C.3	Figure C.4
Five NNs	Figure C.5	Figure C.6	Figure C.7	Figure C.8
Ten NNs	Figure C.9	Figure C.10	Figure C.11	Figure C.12
Twenty NNs	Figure C.13	Figure C.14	Figure C.15	Figure C.16

In general, the inlier ratios obtained by LSH matches with eight reference images are lower than the ones obtained by brute-force matches with a single reference image but in many cases our approach recovers many matches that would have been lost if only the nearest neighbour had been used. For most of the cases, the behaviour of the inlier ratio curves are very similar to the curves obtained by the brute-force matches with a single reference image.

For *Graffiti*, the baseline inlier ratio for the top-ranked 250 BRISK correspondences is 33%. This ratio is increased to 47% and 48% by our approach with bit grouping of 4 and 8 bits respectively which is shown in Figure C.3. With other descriptors, we have also significant improvement in inlier ratios for best few correspondences and also overall inlier ratio. As it is illustrated in Figure C.11, the overall inlier ratio is increased from 13% to 24% with FREAK correspondences with bit groups of 6 or 8.

For *Bikes*, we have an important improvement in inlier ratios with BRISK and FREAK correspondences. For the top-ranked 300 FREAK correspondences, we increase the inlier ratio from 55% to 66% and 68% by groups of 4 bits and 8 bits respectively which is shown in Figure C.1. With BRISK correspondences, the overall inlier ratio is increased from 17% to 25% as it is illustrated in Figure C.9. For other descriptors, we also have improvement both in inlier ratio for the best few hundred correspondences and

the overall inlier ratio but not as much as BRISK and FREAK.

We have significant improvement for the *Boat* with any of the descriptors except BRIEF. For example, among the top-ranked 250 ORB correspondences, almost none of the nearest neighbour matches is inlier. By our approach, 30% of the best 250 correspondences are correctly matched that is shown in Figure C.2. We have also improvement in the overall inlier ratio. As it is illustrated in Figure C.10, with BRISK correspondences, the overall ratio is increased from 7% to 18% by our approach with grouping of 8 bits.

We have great improvement for the *Wall* especially for the best few hundred correspondences. For example, the inlier ratio for the top-ranked 300 FREAK correspondences is increased from 63% to 77% and 80% by our approach with groups of 4 and 8 bits respectively which is shown in Figure C.4. The overall inlier ratios are also improved for all descriptors. For example, the overall inlier ratio is increased from 22% to 27% by 8-bit length groups with LATCH correspondences as it illustrated in Figure C.12.

Table 5.4. 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. Similar to the results of Table 5.2, ranking the correspondences by Equation 5.3 improves the overall values. Searching in the first ten near neighbors further increases the number of inliers that can be obtained.

	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	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	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	.
	5NN-4	0.94	0.91	0.84	0.70	0.52	0.40	0.83	0.71	0.51	0.96	0.96	0.71	0.99	0.97	.
	5NN-8	0.95	0.92	0.87	0.71	0.55	0.40	0.84	0.72	0.52	0.98	0.96	0.71	0.99	0.98	.
	10NN-4	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	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	.
	20NN-4	0.94	0.91	0.83	0.70	0.52	0.41	0.83	0.71	0.51	0.96	0.96	0.71	0.99	0.96	.
	20NN-8	0.95	0.92	0.86	0.70	0.56	0.42	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	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	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
	5NN-4	0.00	0.00	0.00	0.88	0.56	0.32	0.76	0.66	0.44	0.53	0.26	0.14	0.42	0.24	0.14
	5NN-8	0.00	0.00	0.00	0.91	0.58	0.32	0.80	0.70	0.45	0.59	0.27	0.14	0.47	0.26	0.14
	10NN-4	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	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**
	20NN-4	0.00	0.00	0.00	0.89	0.61	0.37	0.76	0.70	0.53	0.63	0.35	0.19	0.49	0.29	0.19
	20NN-8	0.00	0.00	0.00	0.92	0.66	0.39	0.82	0.76	0.56	0.67	0.37	0.20	0.54	0.33	0.19
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	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	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
	5NN-4	0.36	0.24	0.18	0.66	0.45	0.29	0.67	0.48	0.33	0.44	0.26	0.16	0.42	0.31	0.20
	5NN-8	0.43	0.28	0.19	0.70	0.47	0.30	0.68	0.51	0.35	0.46	0.28	0.17	0.46	0.32	0.20
	10NN-4	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	**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**
	20NN-4	0.35	0.26	0.22	0.66	0.47	0.31	0.67	0.49	0.35	0.42	0.27	0.17	0.42	0.30	0.20
	20NN-8	0.50	0.38	0.27	0.70	0.48	0.33	0.68	0.53	0.39	0.49	0.29	0.18	0.46	0.34	0.22
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	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	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
	5NN-4	1.00	0.96	0.86	**1.00**	0.95	0.66	0.96	0.82	0.58	0.90	0.83	0.50	0.98	0.93	0.58
	5NN-8	1.00	0.98	0.89	**1.00**	0.97	0.67	0.95	0.83	0.59	0.92	0.85	0.51	0.99	0.94	0.59
	10NN-4	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	**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**
	20NN-4	0.99	0.96	0.86	**1.00**	0.95	0.66	0.96	0.81	0.59	0.90	0.84	0.52	0.98	0.92	0.59
	20NN-8	1.00	0.98	0.88	**1.00**	0.97	0.68	0.95	0.84	0.61	0.92	0.86	0.53	0.99	0.94	0.59

5.4. Discussion

As it can be seen from the Figure 5.11 - 5.14, we have significant improvement in recognition rate except some cases. The improvement is less pronounced for the *Bikes* sequence with only changes in blur level. This is expected since the observation probabilities represent behaviour 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. The BRIEF descriptor lacks orientation estimation, as a result there is no improvement after the test image 2 of the *Boat* sequence. This indicates that even the ten near neighbour list does not include enough correct matches. For the other descriptors with orientation estimation, the results are improved by a large amount for the test image 3 and 4.

Reweighted nearest neighbour curves which are illustrated in Figure B.1 - B.4 show that keypoint specific scoring brings the inliers closer to the top of the ranked correspondence list. This provides to have higher inlier ratios at the level of first several hundred correspondences without increasing the final inlier count. Algorithms like [9] use a few top-ranked correspondences for keypoint matching and keypoint specific scoring might provide to have more correct matches in this small set of best correspondences. Our approach has also significant improvement in the overall inlier ratios in most of the cases. This means that more correct matches can be obtained between the different views of the scenes belong to a single image.

The number of inliers that can be obtained is increased by our approach even we compute the near neighbour list with the LSH as it is illustrated in Figure C.9 - C.12. This is important since the reference image collection is larger than the single image for some vision applications and approximate nearest neighbour methods are used to find possible matches. It can be stated that our approach scales well to larger reference descriptor collections since we only compute keypoint specific score for the top ten near neighbours instead of all reference keypoints and these match hypotheses can be efficiently computed with approximate nearest neighbour methods.

As it is stated in the Section 5.3.2, the difference is negligible for timing between the brute-force search time for the nearest neighbour and establishing correspondences with our approach. Therefore, computing the ten near neighbour list and reranking these top-ranked matches according to keypoint specific and probabilistic score is both computationally efficient and provides better keypoint matching. It can be seen from the all

experiments, our approach is relatively descriptor independent and it extends the matching range of several binary descriptors beyond the nearest neighbour.

5.5. Conclusion

Binary descriptors are widely used for keypoint matching in real-time vision applications. The common approach for matching keypoints is to find the nearest neighbour in the set of descriptors according to Hamming distance. Perspective changes cause specific variations in descriptor of each keypoint and this leads to obtain incorrect matches with the nearest neighbour.

For better keypoint matching, we first identify the list of the top ten near neighbours according to the Hamming distance. These matches are then reranked based on the keypoint specific and probabilistic match quality score obtained from the texture characteristics around individual keypoints. This two-step approach provides to have more correct matches regardless of type of the binary descriptor. Our approach scales well to large descriptor sets with approximate nearest neighbour search algorithms and also represent each keypoint with a robust and distinctive description.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1. Conclusion

In this thesis, we proposed an approach that can be used in conjunction with the binary descriptors to search for keypoint matches beyond the nearest neighbour. We showed that binary descriptors are not fully invariant to perspective changes and variations in bits are dependent on texture in the patch surrounding to keypoint. We also showed that correct matches which cannot be found by searching the nearest neighbour is more likely to be in list of top K near neighbours. By combining these properties, we developed an approach for keypoint matching with binary descriptors.

We proposed a training procedure that covers various deformations by simulating different viewpoints which are close to the real scenes that provides to exploit the texture characteristics around individual keypoints in an off-line training phase. We captured these keypoint specific descriptor variations with probabilistic conditional model. In order to make our approach feasible, we split the binary descriptor vector into groups of bits and assumed that these groups are independent from each other instead of accepting independence of each bit which leads to have a large number of parameters to represent the model.

We developed a two step approach for keypoint matching. For each given query keypoint, we first computed the top K near neighbours based on the Hamming distance with a negligible difference in time which is needed to search only the nearest neighbour. We then reranked keypoints in the top K near neighbours list by targeting to bring the correct match to the nearest neighbour with a keypoint specific and probabilistic score. We computed this score by combining the Hamming distance and the probability of the observing the query descriptor which is obtained from the probabilistic model.

Our approach increases the robustness of state-of-the-art binary descriptors under most of the affine deformations. With BRIEF descriptors, there is no improvement under high level of rotations since there is no orientation estimation mechanism. For scale and

tilt variations, we have important improvement especially for the images that have both scale and tilt changes with a less amount rotation. BRISK and FREAK show similar behaviour. We extend the matching range of them for all type of affine deformations and blur changes. We have significant improvement with BRISK and FREAK especially under scale and rotation changes. Our approach also increases the robustness of the LATCH and ORB descriptors under rotation,scale, and tilt deformations. Since we extract texture characteristics by simulating real scenes on an image from different viewpoints, we have great improvement in recognition performance for rotation, scale, and tilt changes. For blur changes, there is no improvement with BRIEF, LATCH, and ORB descriptors since observation probabilities that we compute in an offline phase represents bit variations under only affine deformations.

Our approach strikes a balance between the keypoint specific representations and the generic keypoint descriptors. The former approaches exploit the texture characteristics around individual keypoints to yield a more robust and distinctive description. The latter ones scale well to large data sets thanks to the efficient approximate neighbour search algorithms. By using a two step approach, we combine the advantages of both and improve the robustness of binary descriptors for real-time vision applications.

6.2. Future Work

The idea of searching the match in the list of K near neighbours is rarely exploited in practice. This approach can be combined with different keypoint matching approaches especially for the ones that use keypoint specific representations for better matching.

Observation probabilities which are obtained in an off-line training phase are needed to be stored and this may be problematic for some applications due to memory concerns. Instead of simulating thousand number of different viewpoints and observing descriptor variations in an off-line training phase, extraction of observation probabilities with a few number of synthetic images at run-time might make our approach feasible to more vision applications.

We have achieved good matching performance with binary descriptors for 2D planar objects and we would like to apply our approach to the 3D objects. With some adaptations such as taking into account epipolar constraints, our approach might provide robust keypoint matching.

REFERENCES

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. pages 510–517, 2012.

[2] M. Alzantot and M. Youssef. Uptime: Ubiquitous pedestrian tracking using mobile phones. In *WCNC*, pages 3204–3209. IEEE, 2012.

[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[4] V. Balntas, L. Tang, and K. Mikolajczyk. Bold-binary online learned descriptor for efficient image matching. pages 2367–2375, 2015.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. 10(3):346–359, 2008.

[6] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision*, 74(1):59–73, Aug. 2007.

[7] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. BRIEF: Computing a Local Binary Descriptor Very Fast. 34(7):1281–1298, 2012.

[8] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, J. Singh, and B. Girod. Transform coding of image feature descriptors, 2009.

[9] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. pages 220–226, San Diego, CA, June 2005.

[10] O. Chum, A. Mikulik, M. Perdoch, and J. Matas. Total recall ii: Query expansion revisited. pages 889–896, June 2011.

[11] P. Ghosh, S. Antani, L. R. Long, and G. R. Thoma. Review of medical image retrieval systems and future directions. In *Proceedings of the 2011 24th International Symposium on Computer-Based Medical Systems*, CBMS '11, pages 1–6, Washington,

DC, USA, 2011. IEEE Computer Society.

[12] R. Gupta and A. Mittal. Smd: A locally stable monotonic change invariant feature descriptor. pages 265–277. 2008.

[13] H. Kawaji, K. Hatada, T. Yamasaki, and K. Aizawa. Image-based indoor positioning system: Fast image matching using omnidirectional panoramic images. In *Proceedings of the 1st ACM International Workshop on Multimodal Pervasive Video Analysis*, MPVA '10, pages 1–4, New York, NY, USA, 2010. ACM.

[14] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'04, pages 506–513, Washington, DC, USA, 2004. IEEE Computer Society.

[15] V. Lepetit and P. Fua. Keypoint recognition using Randomized Trees. 28(9):1465–1479, Sept. 2006.

[16] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. pages 2548–2555, 2011.

[17] G. Levi and T. Hassner. LATCH: learned arrangements of three patch codes. *CoRR*, abs/1501.03719, 2015.

[18] X. Li, M. Larson, and A. Hanjalic. Pairwise geometric matching for large-scale object retrieval. June 2015.

[19] D. Lowe. Distinctive image features from scale-invariant keypoints. 20(2):91–110, 2004.

[20] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings of the European Conference on Computer Vision (ECCV'10)*, September 2010.

[21] A. S. Mian, M. Bennamoun, and R. Owens. Keypoint detection and local feature

matching for textured 3d face recognition. *International Journal of Computer Vision*, 79(1):1–12, 2008.

[22] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. 27(10):1615–1630, 2004.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. 65(1/2):43–72, 2005.

[24] J.-M. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[25] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. 36, 2014.

[26] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. 2006.

[27] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast Keypoint Recognition Using Random Ferns. 32(3):448–461, 2010.

[28] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. 32(1):105–119, Jan 2010.

[29] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 2011.

[30] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual slam. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2352–2359, Washington, DC, USA, 2011. IEEE Computer Society.

[31] T. Trzcinski, M. Christoudias, and V. Lepetit. Learning image descriptors with boosting. 37(3):597–610, March 2015.

[32] T. Trzcinski and V. Lepetit. Efficient discriminative projections for compact binary descriptors. 2012.

[33] H. B. Zaman, P. Robinson, A. F. Smeaton, T. K. Shih, S. A. Velastin, A. Jaafar, and N. M. Ali, editors. *Advances in Visual Informatics - 4th International Visual Informatics Conference, IVIC 2015, Bangi, Malaysia, November 17-19, 2015, Proceedings*, volume 9429 of *Lecture Notes in Computer Science*. Springer, 2015.

APPENDIX A

RECOGNITION RATE EXPERIMENTS WITH FIVE AND TWENTY NEAR NEIGHBOURS

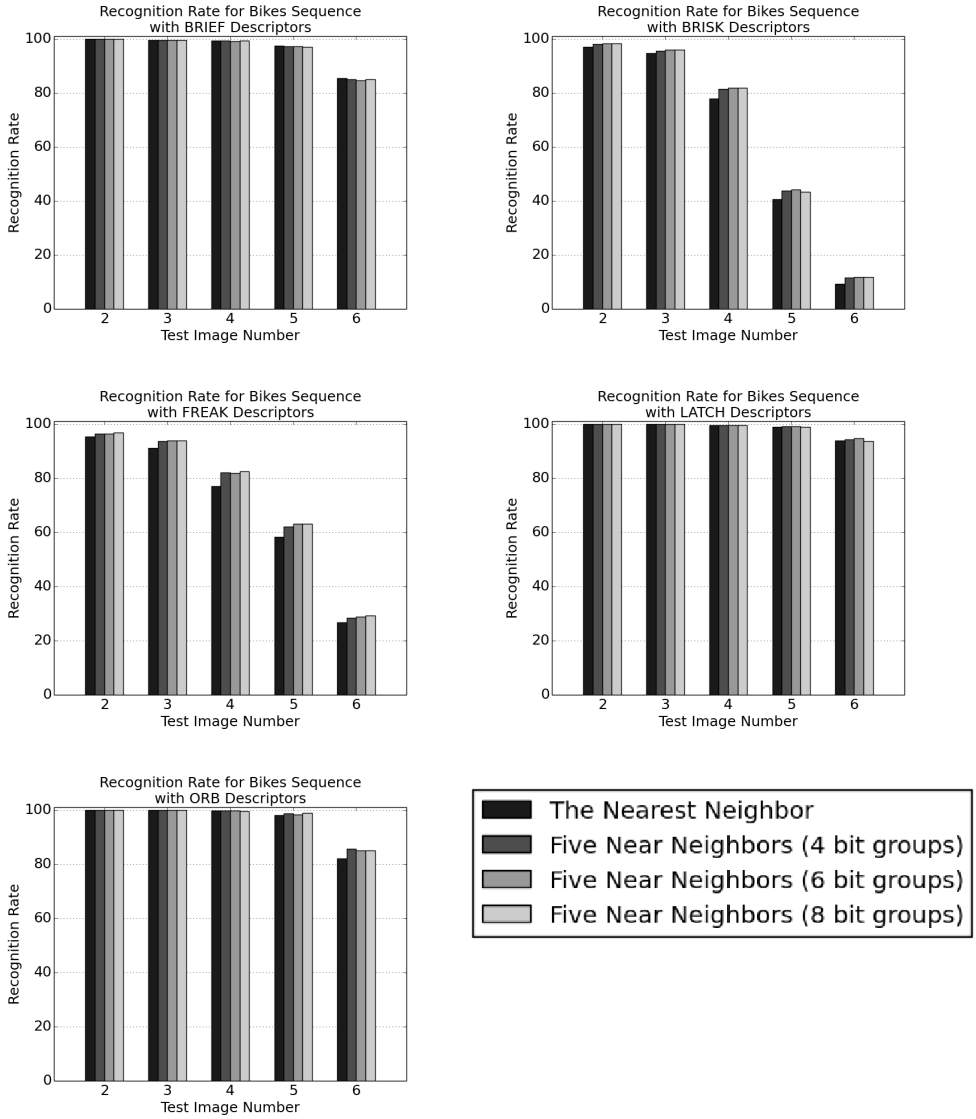


Figure A.1. Recognition rates for the Bikes sequence with various descriptors obtained by using just the nearest neighbour and by ranking the five near neighbours with keypoint specific scoring (Equation 5.3).

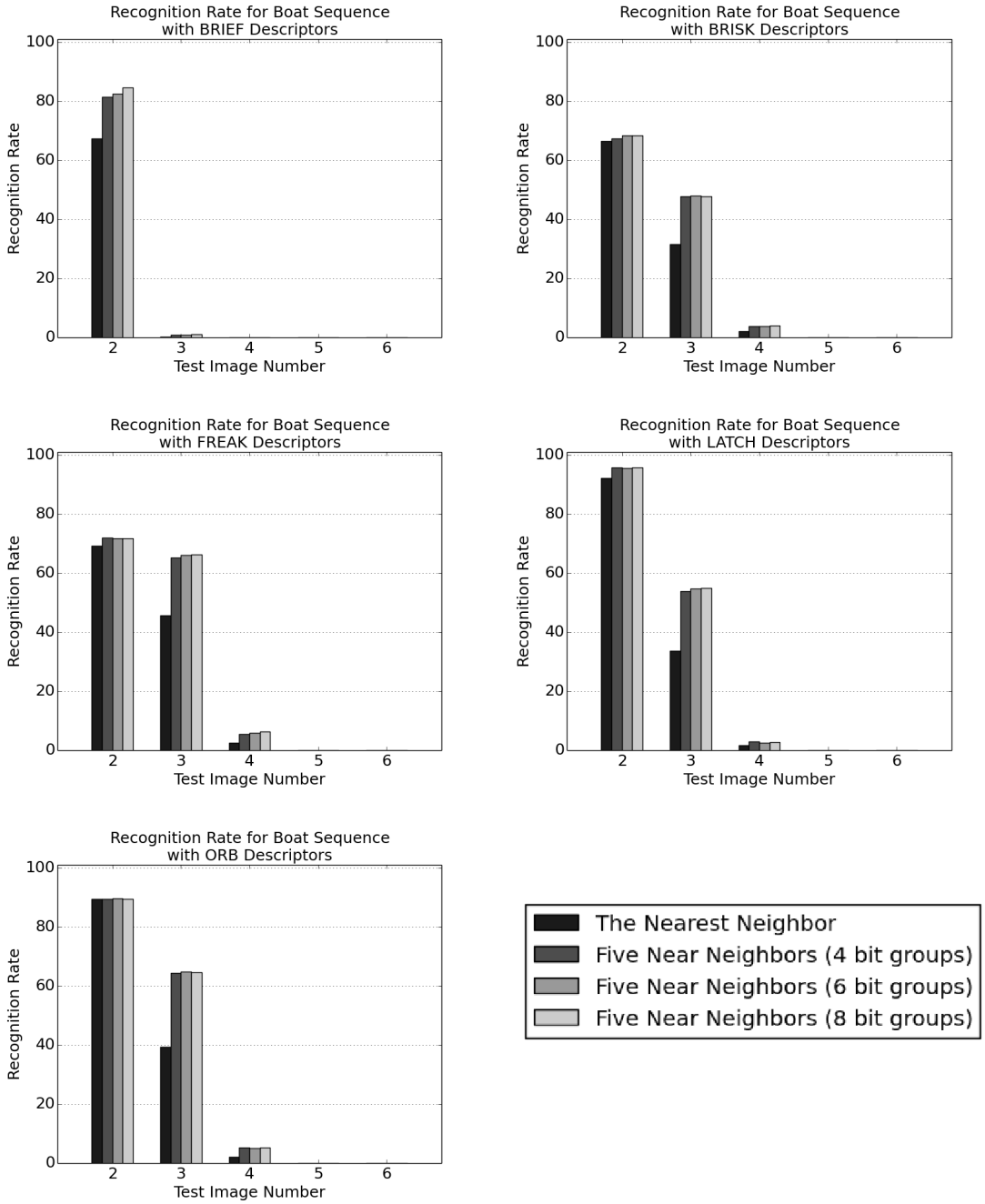


Figure A.2. Recognition rates for the Boat sequence with various descriptors obtained by using just the nearest neighbour and by ranking the five near neighbours with keypoint specific scoring (Equation 5.3).

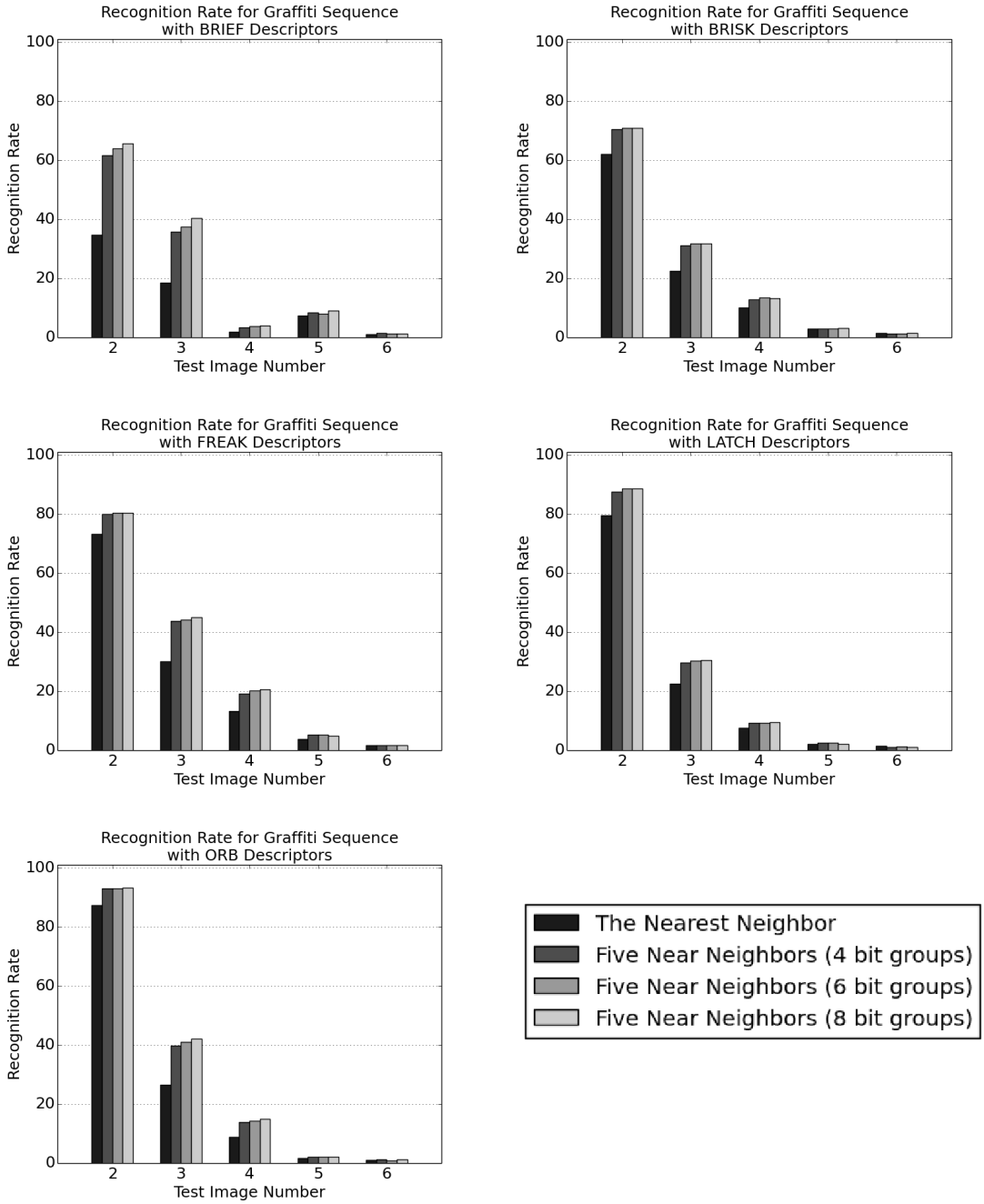


Figure A.3. Recognition rates for the Graffiti sequence with various descriptors obtained by using just the nearest neighbour and by ranking the five near neighbours with keypoint specific scoring (Equation 5.3).

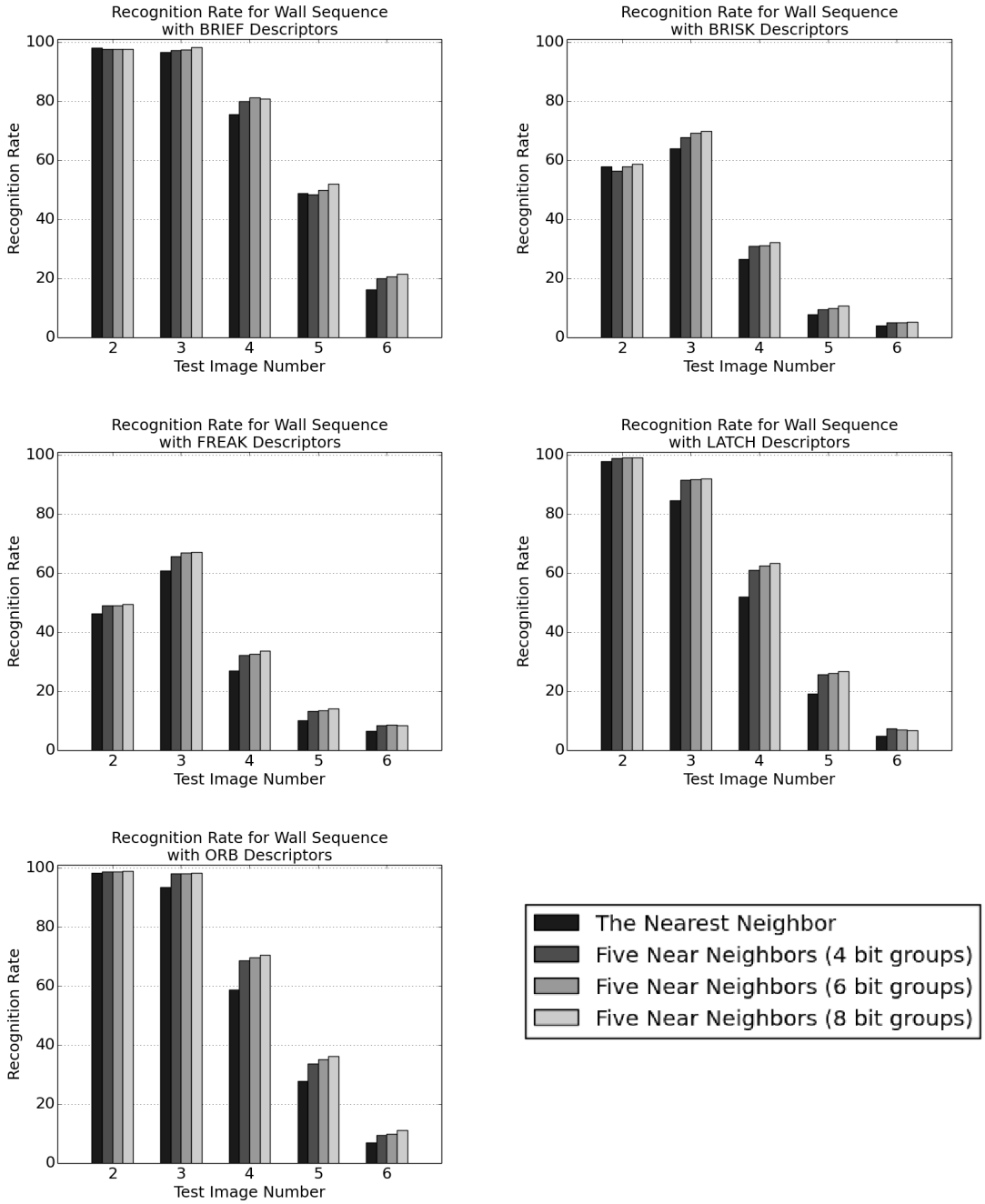


Figure A.4. Recognition rates for the Wall sequence with various descriptors obtained by using just the nearest neighbour and by ranking the five near neighbours with keypoint specific scoring (Equation 5.3).

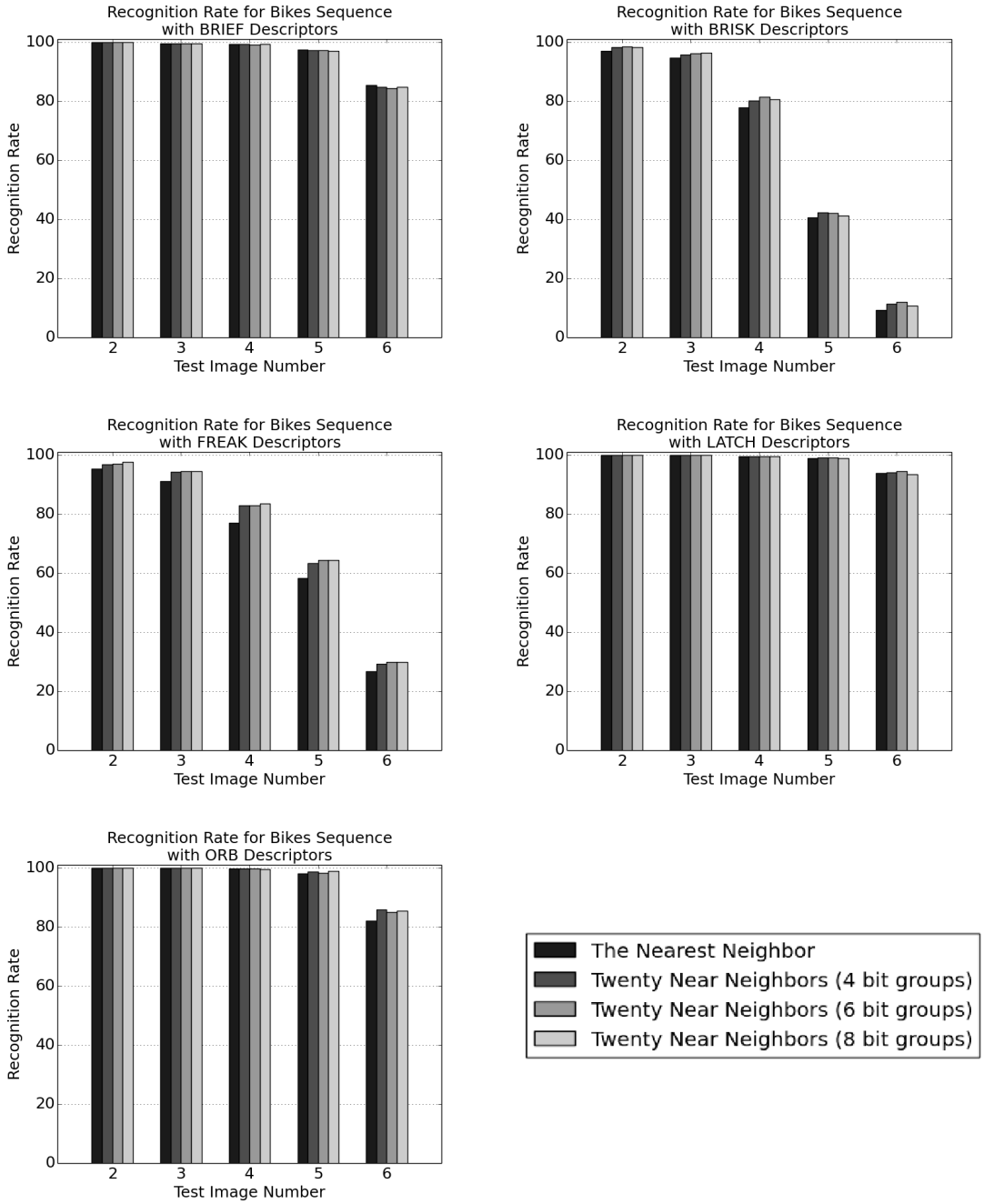


Figure A.5. Recognition rates for the Bikes sequence with various descriptors obtained by using just the nearest neighbour and by ranking the twenty near neighbours with keypoint specific scoring (Equation 5.3).

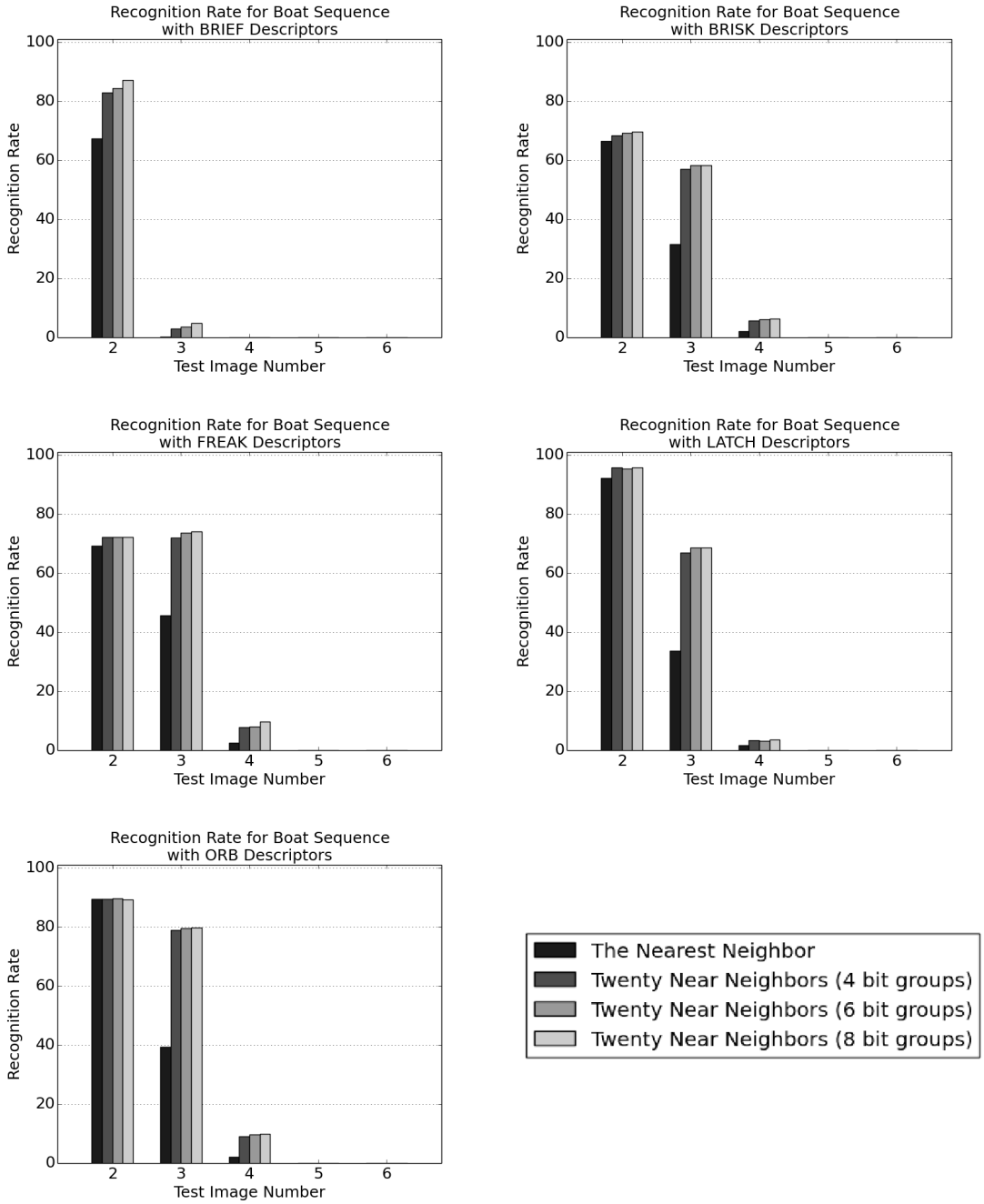


Figure A.6. Recognition rates for the Boat sequence with various descriptors obtained by using just the nearest neighbour and by ranking the twenty near neighbours with keypoint specific scoring (Equation 5.3).

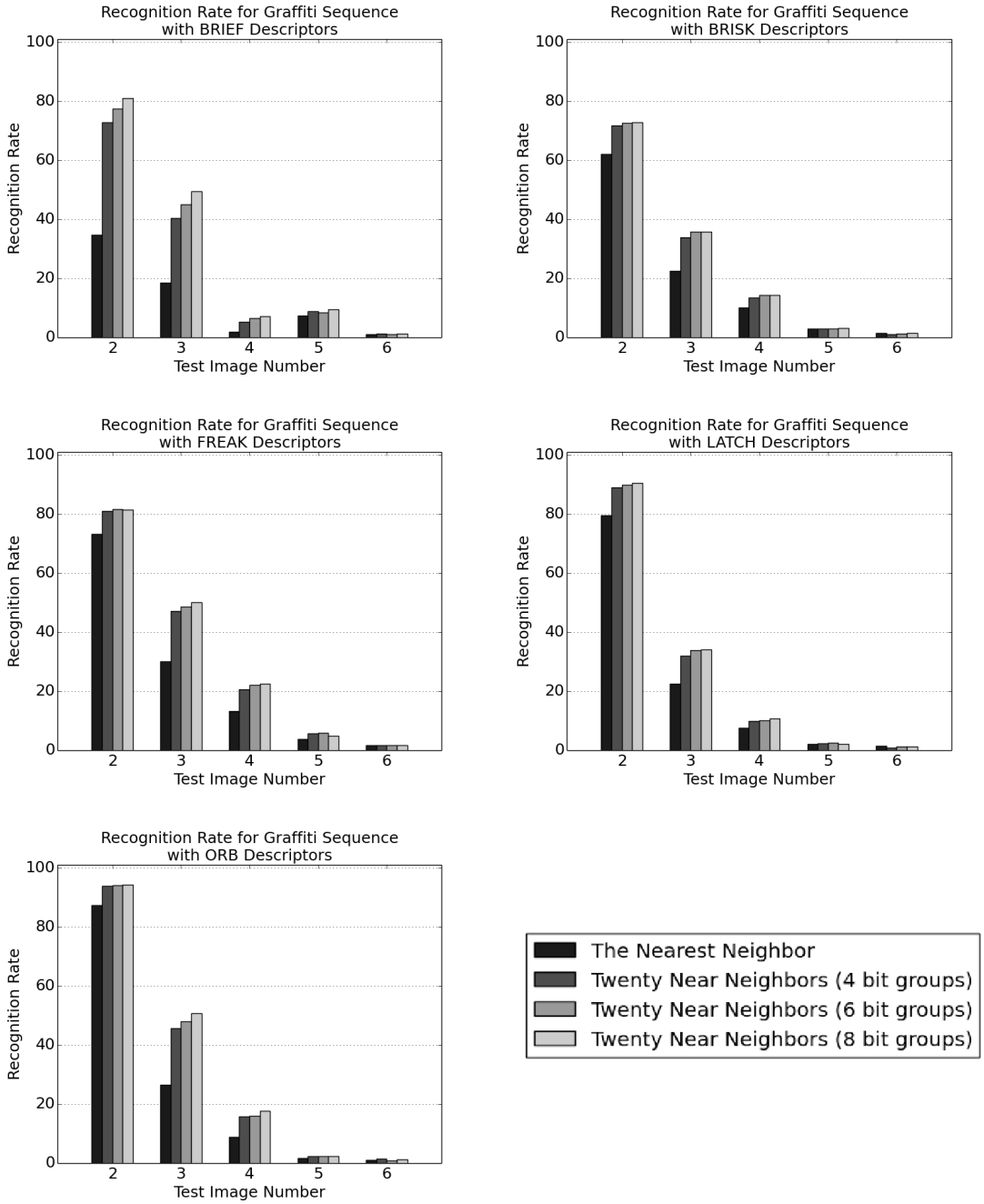


Figure A.7. Recognition rates for the Graffiti sequence with various descriptors obtained by using just the nearest neighbour and by ranking the twenty near neighbours with keypoint specific scoring (Equation 5.3).

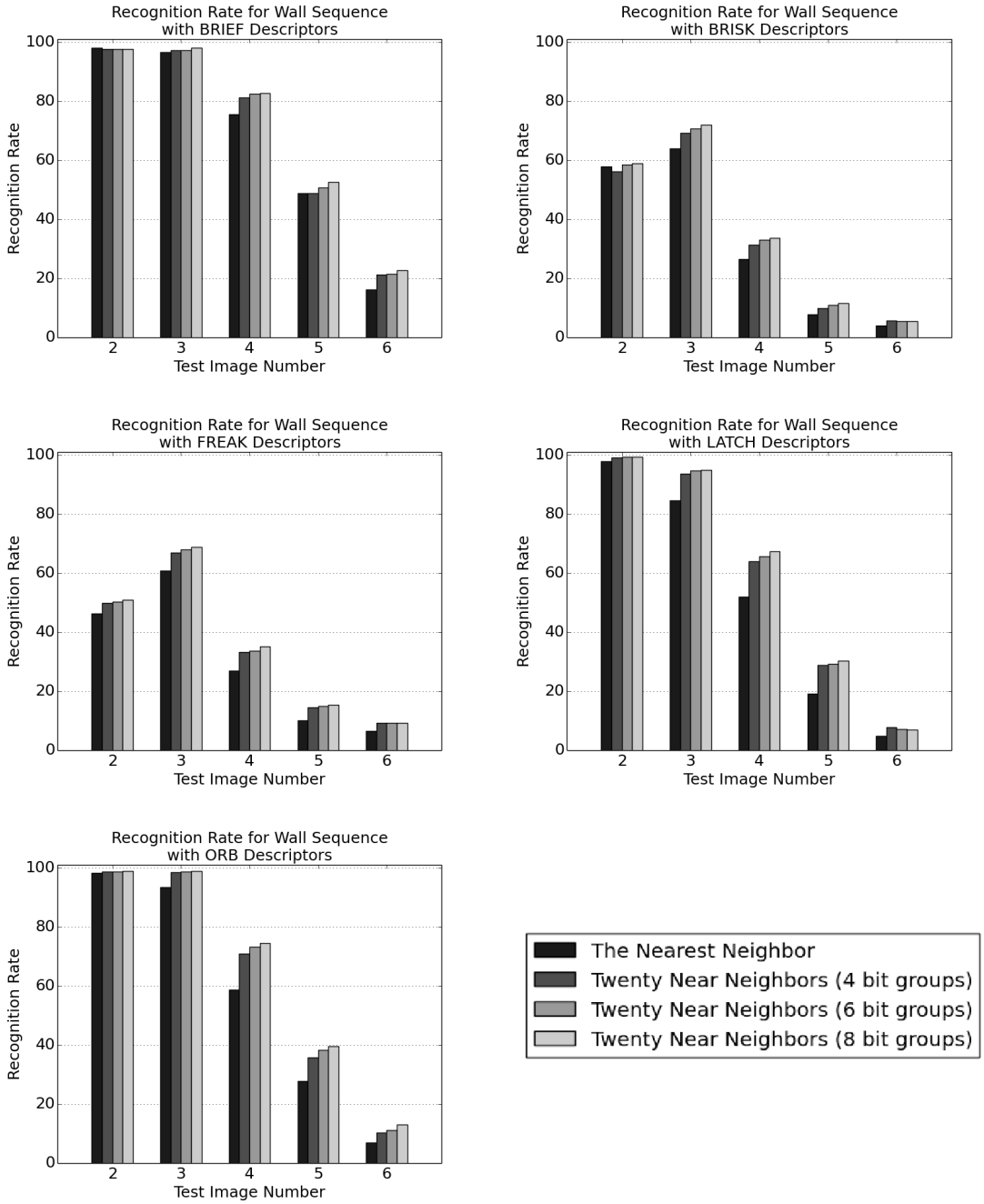


Figure A.8. Recognition rates for the Wall sequence with various descriptors obtained by using just the nearest neighbour and by ranking the twenty near neighbours with keypoint specific scoring (Equation 5.3).

APPENDIX B

INLIER RATIO EXPERIMENTS BY USING BRUTE-FORCE MATCHES

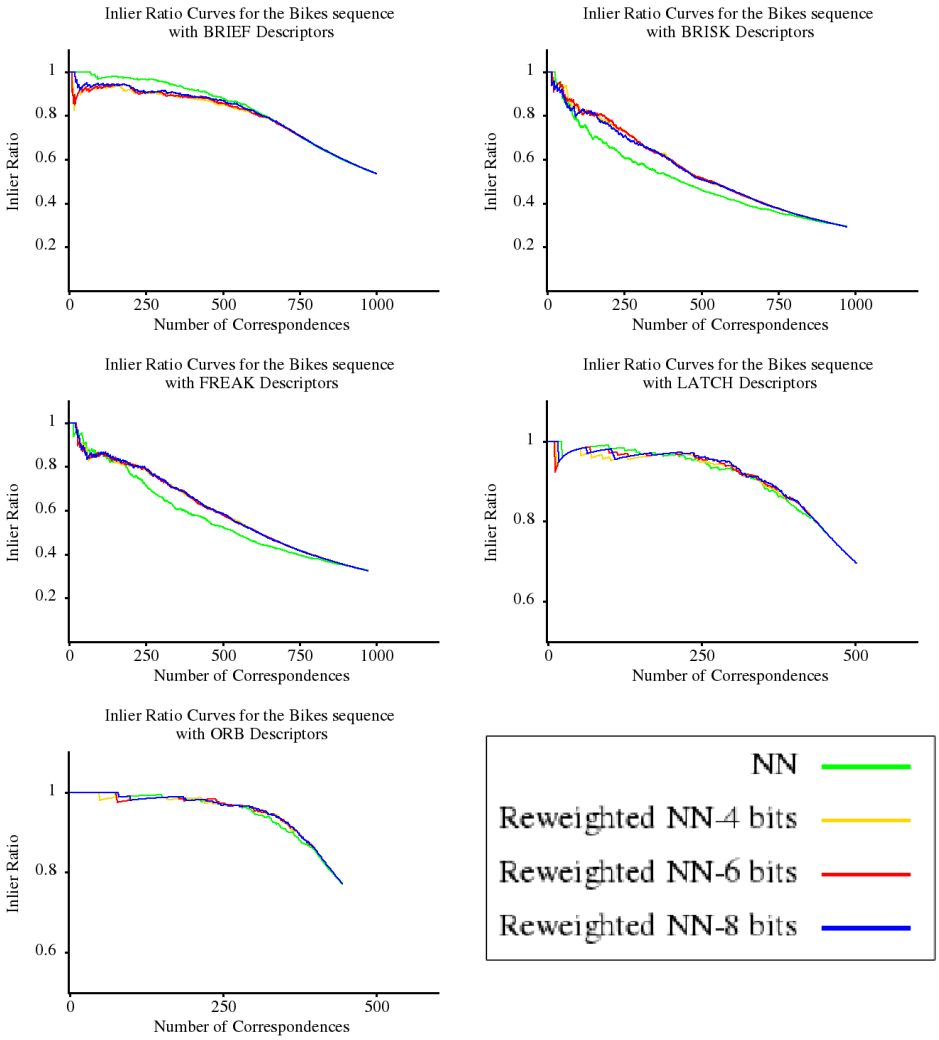


Figure B.1. Inlier ratio curves for the third test image of the Bikes sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

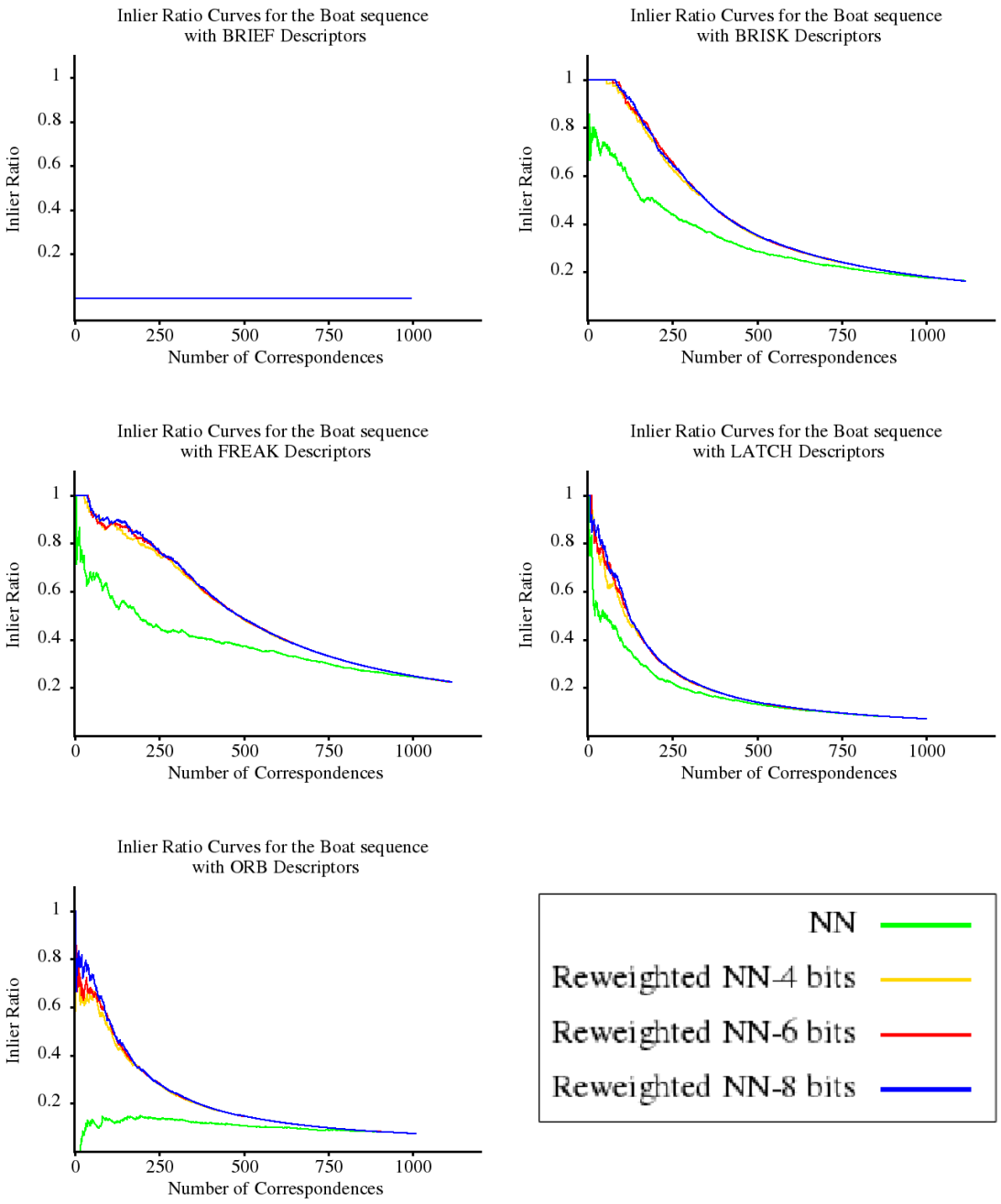


Figure B.2. Inlier ratio curves for the third test image of the Boat sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

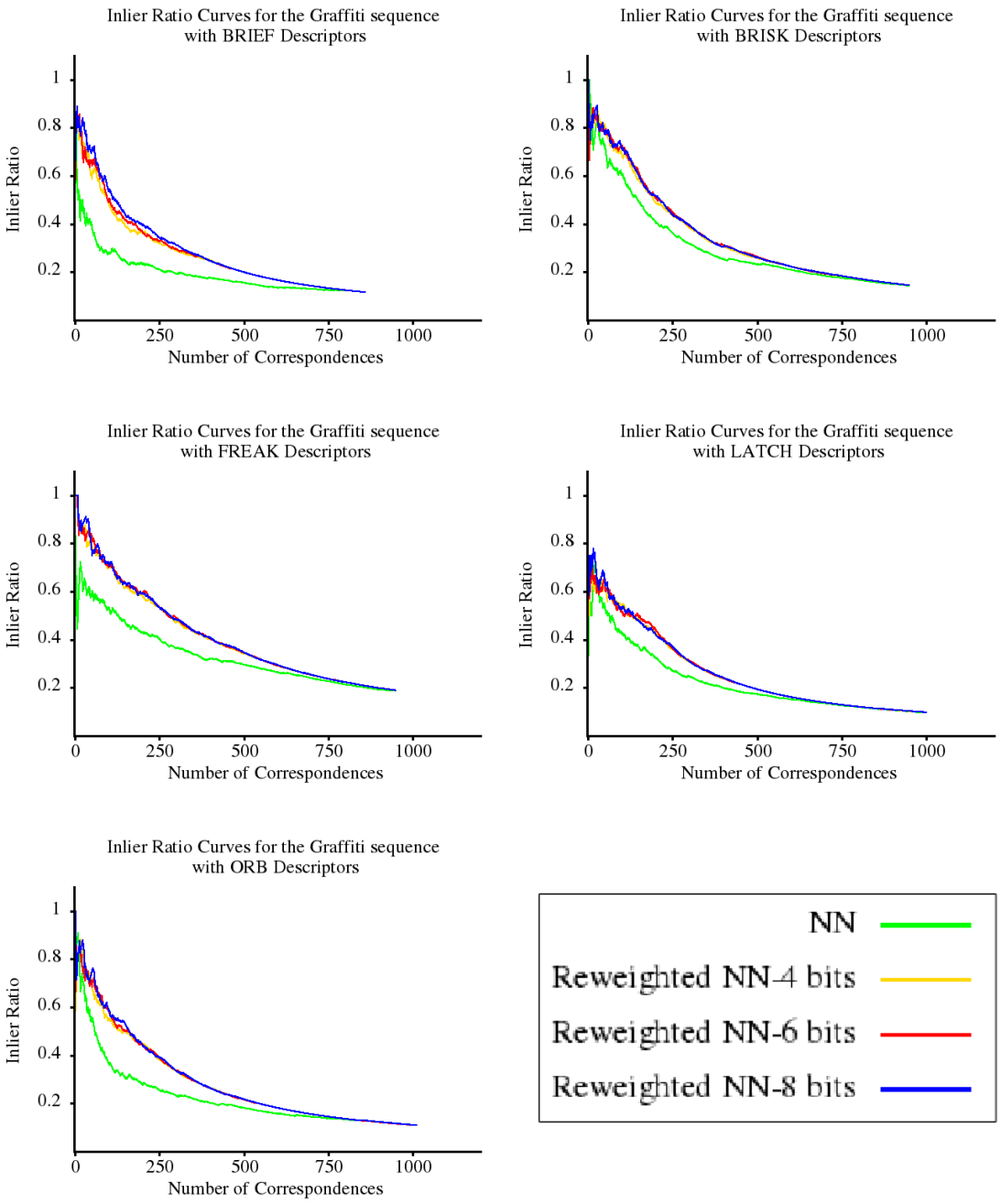


Figure B.3. Inlier ratio curves for the third test image of the Graffiti sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

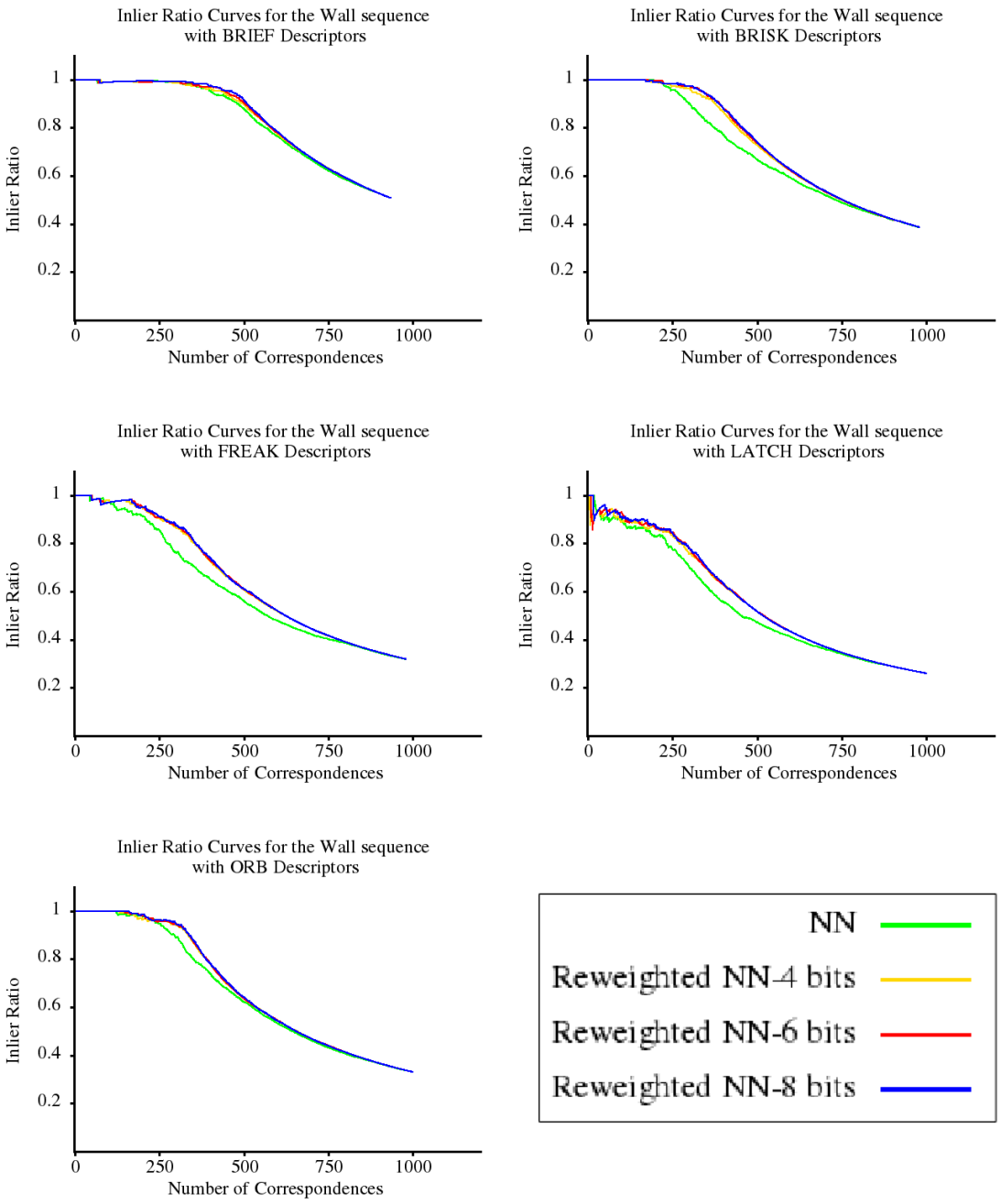


Figure B.4. Inlier ratio curves for the third test image of the Wall sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

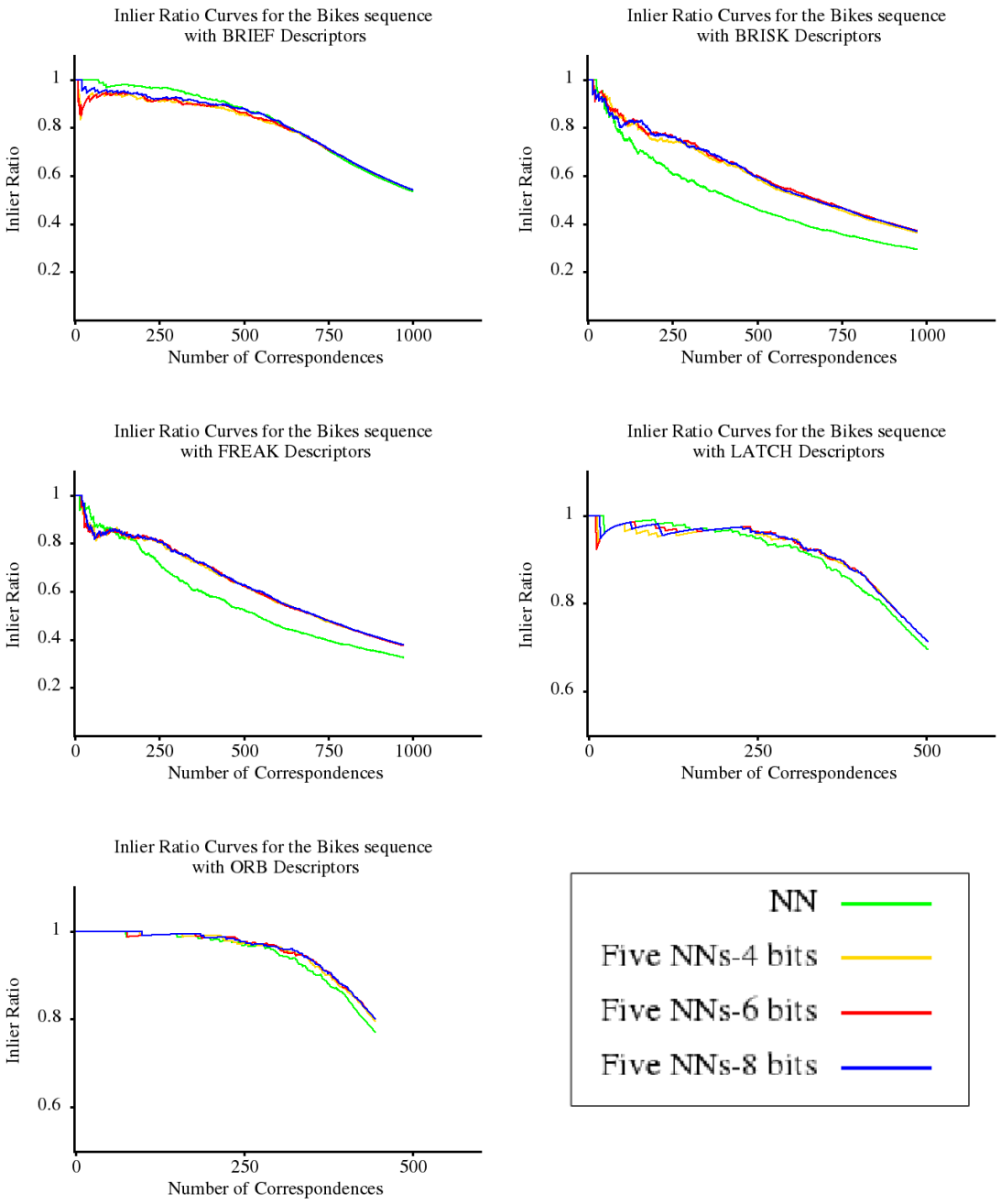


Figure B.5. Inlier ratio curves for the third test image of the Bikes sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

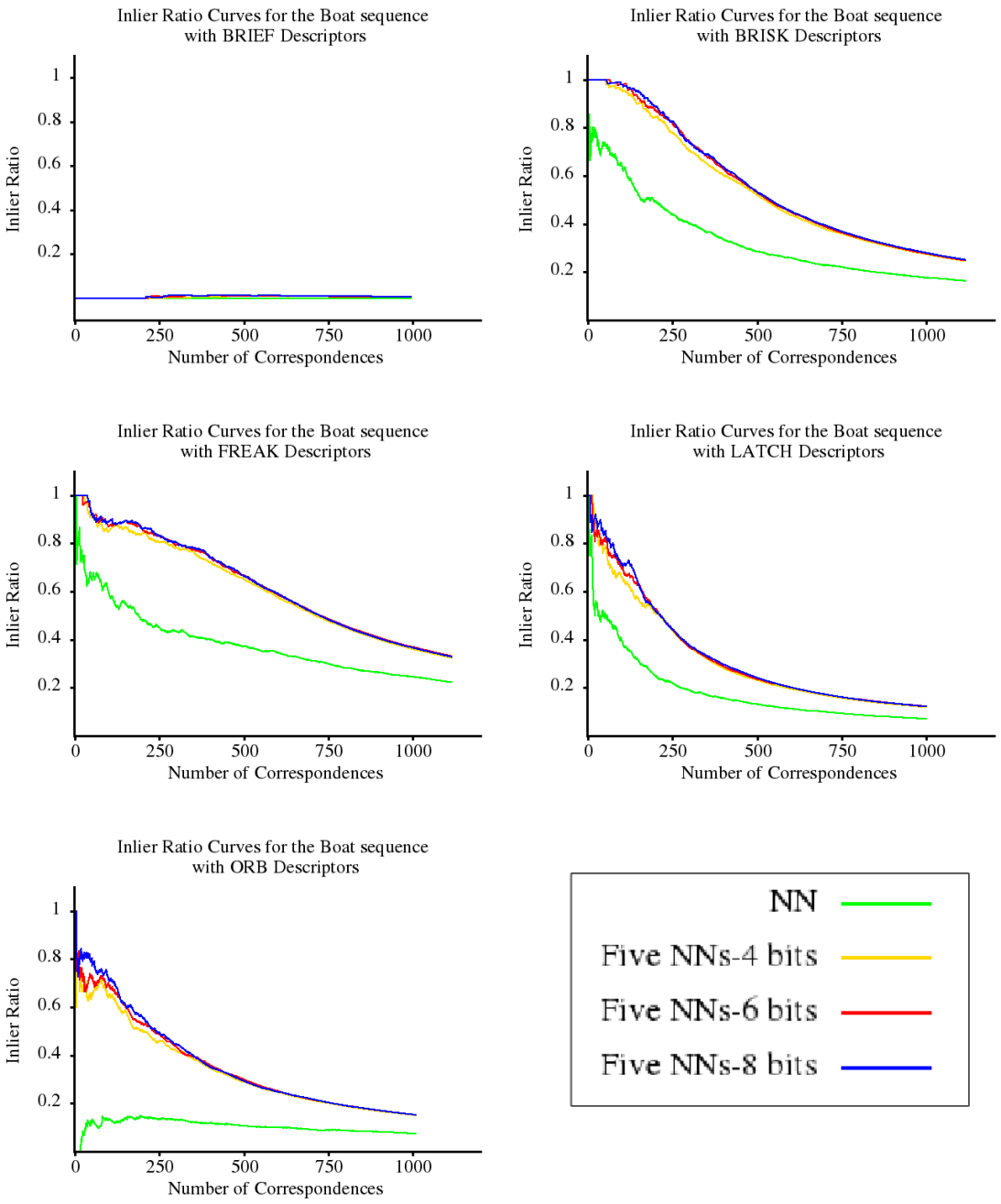


Figure B.6. Inlier ratio curves for the third test image of the Boat sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

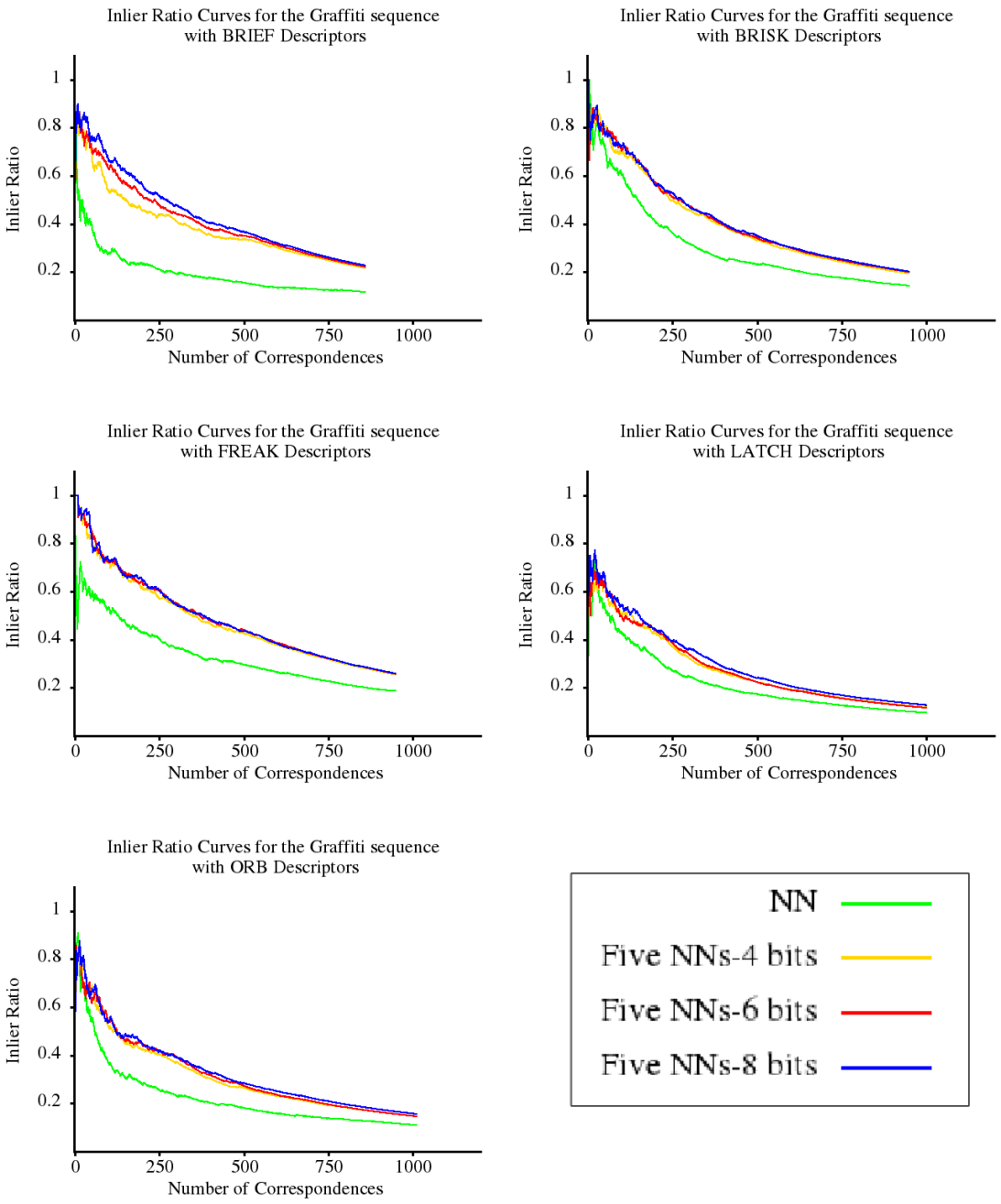


Figure B.7. Inlier ratio curves for the third test image of the Graffiti sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

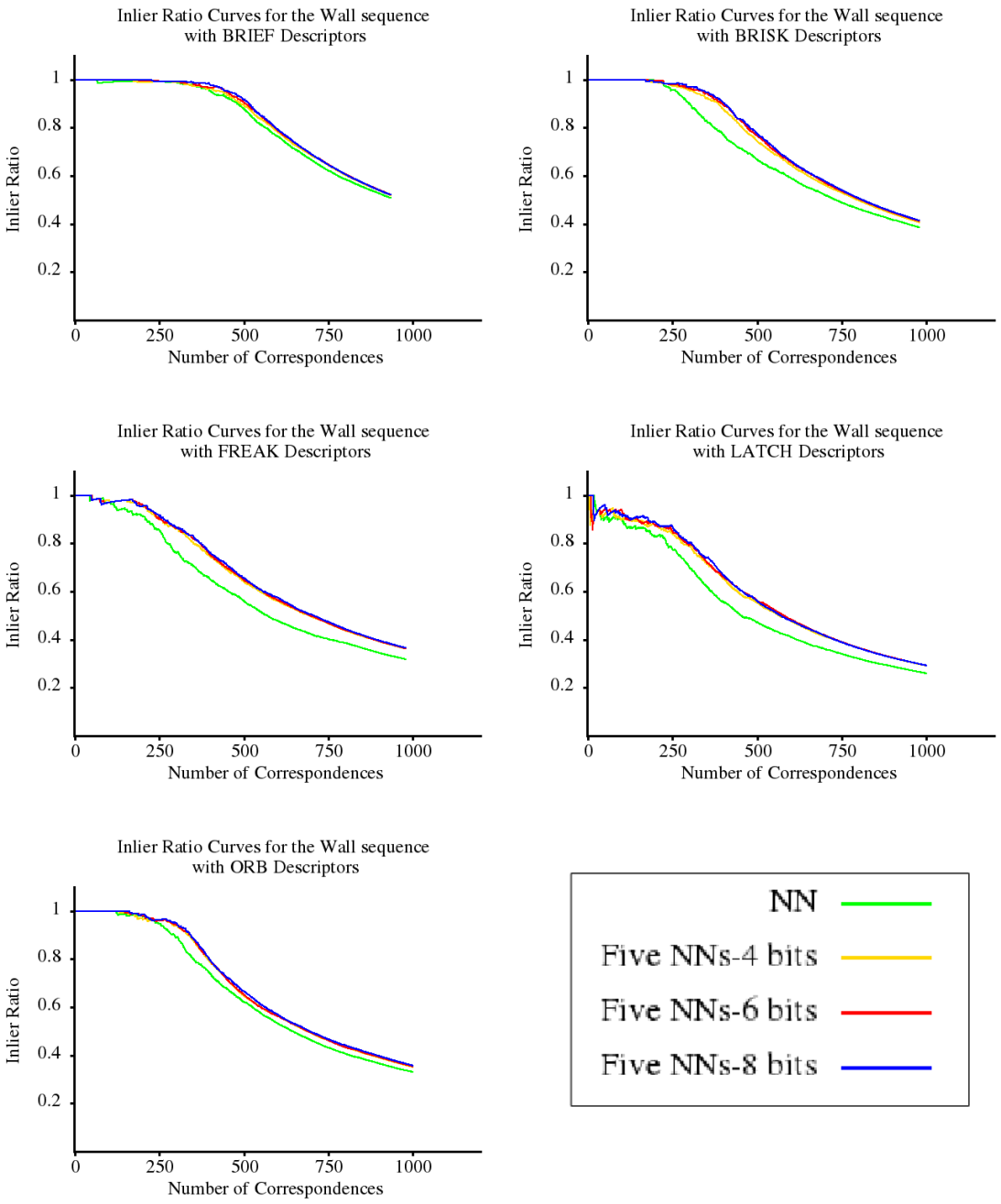


Figure B.8. Inlier ratio curves for the third test image of the Wall sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

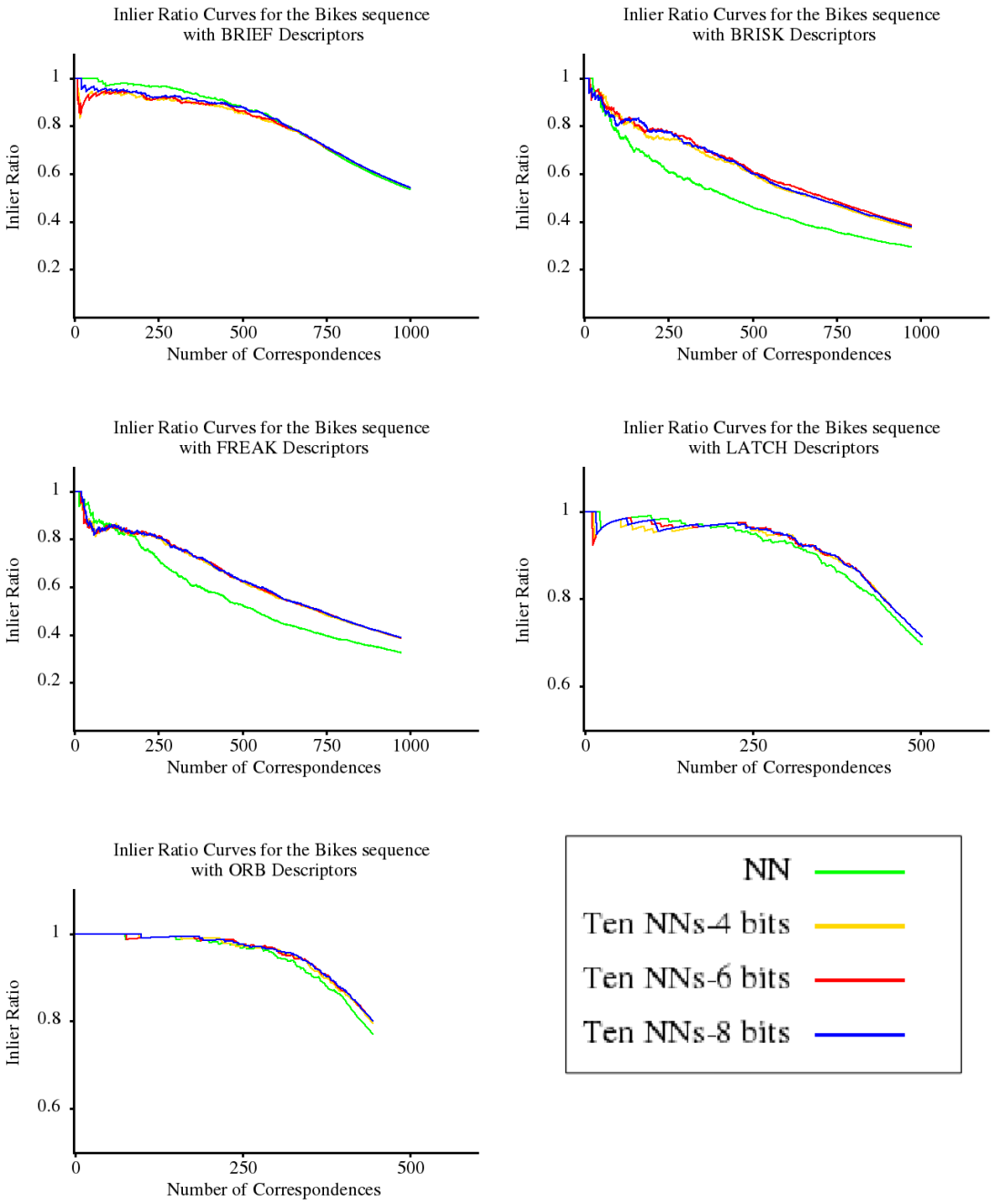


Figure B.9. Inlier ratio curves for the third test image of the Bikes sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

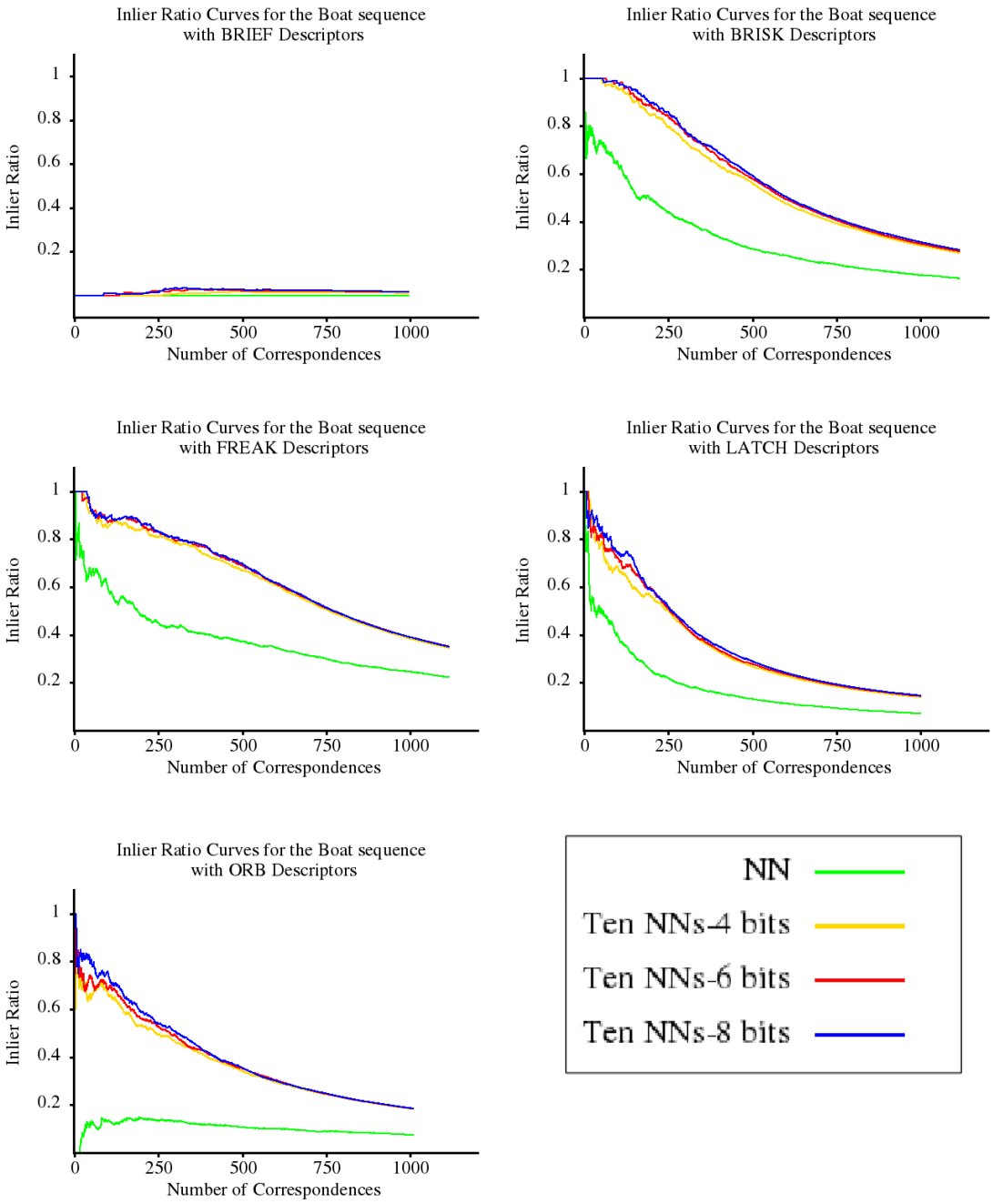


Figure B.10. Inlier ratio curves for the third test image of the Boat sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

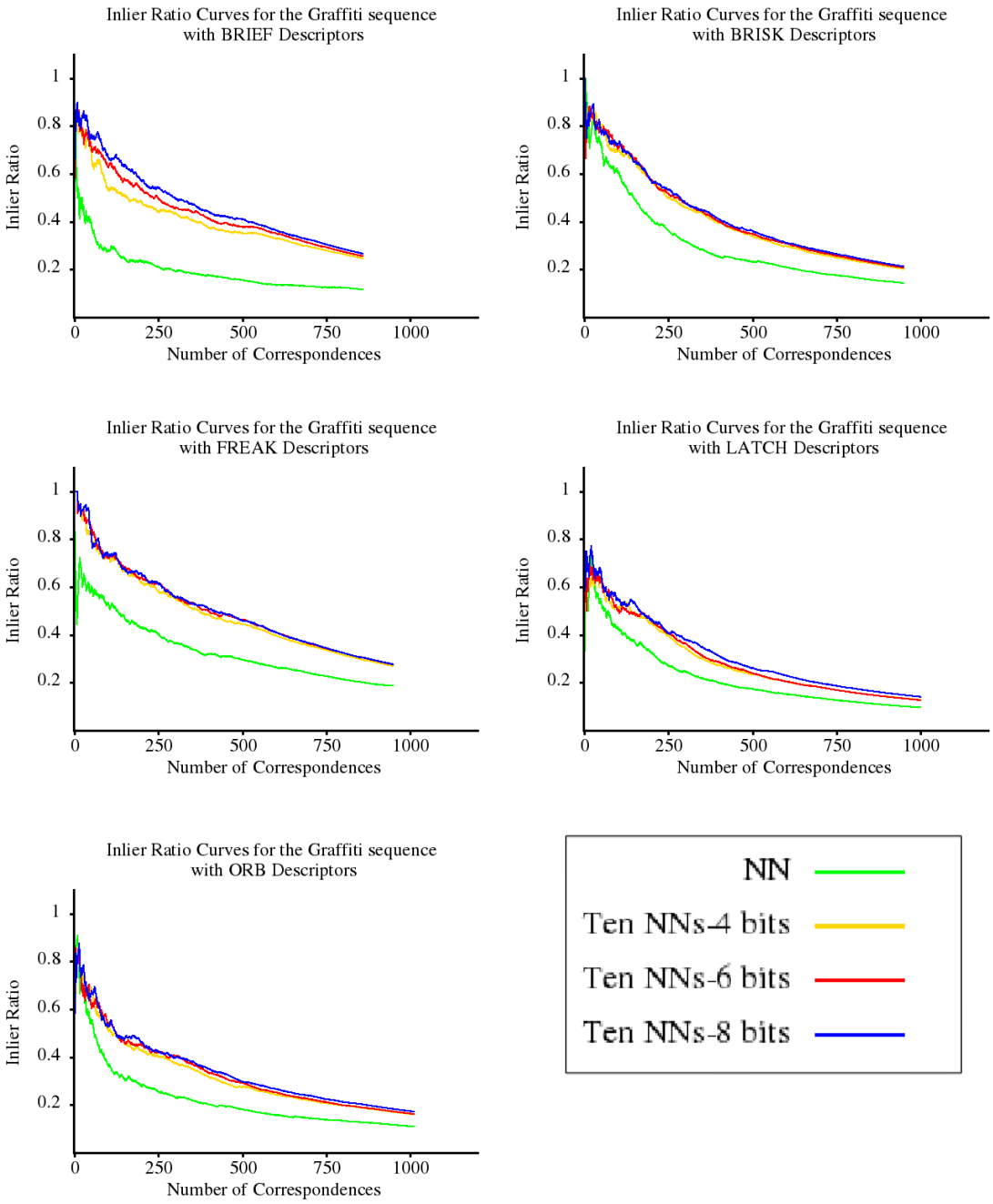


Figure B.11. Inlier ratio curves for the third test image of the Graffiti sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

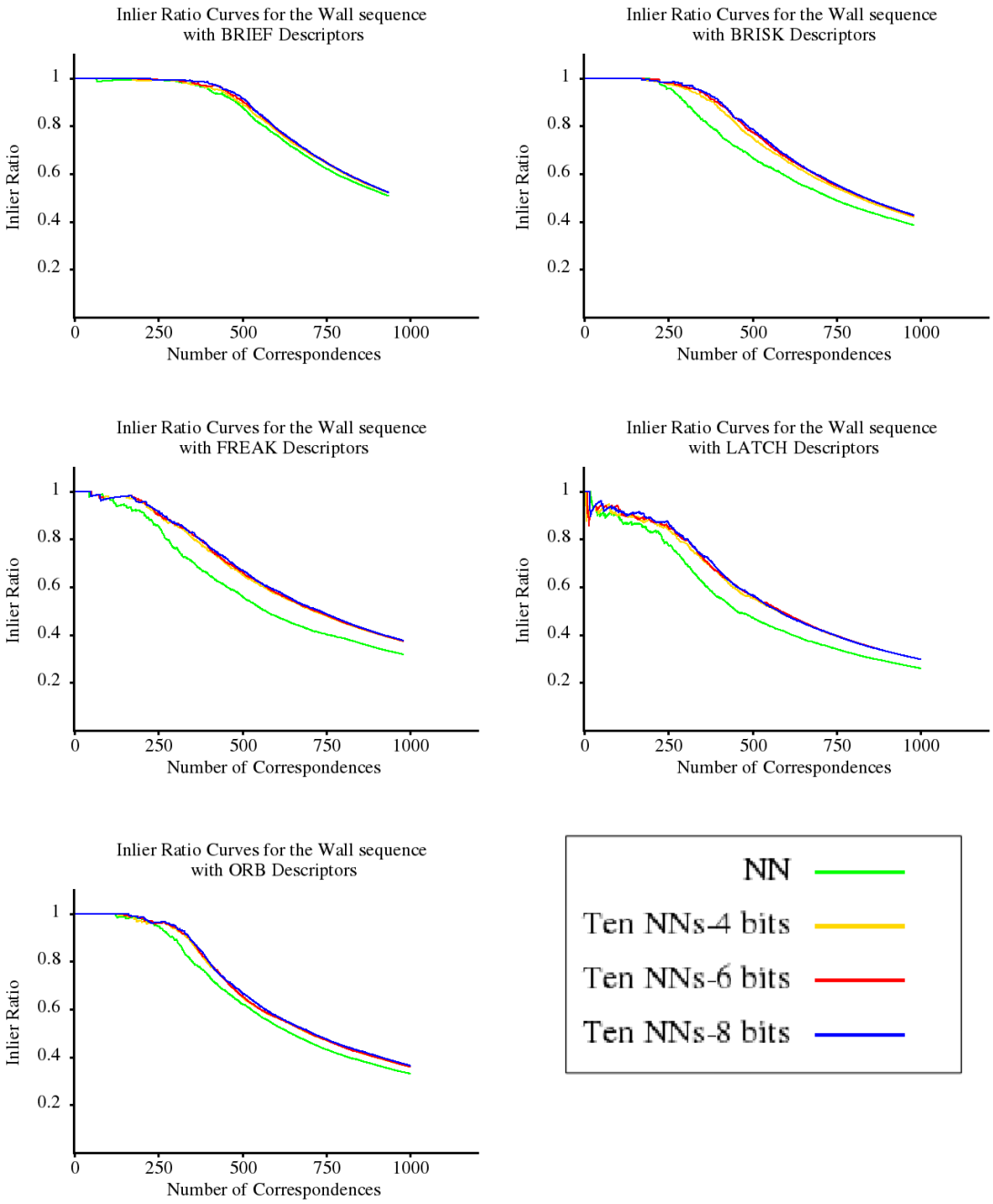


Figure B.12. Inlier ratio curves for the third test image of the Wall sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

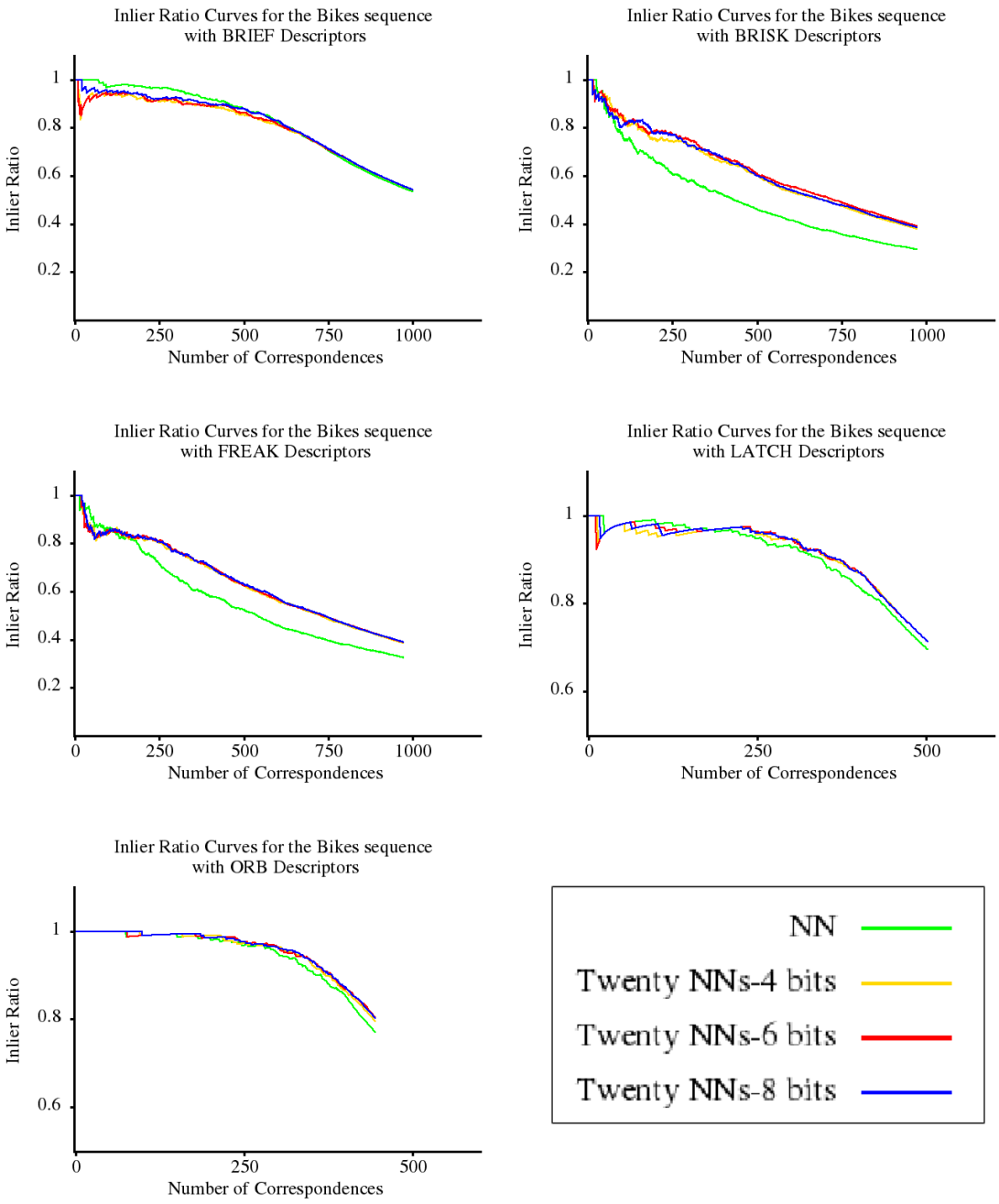


Figure B.13. Inlier ratio curves for the third test image of the Bikes sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.

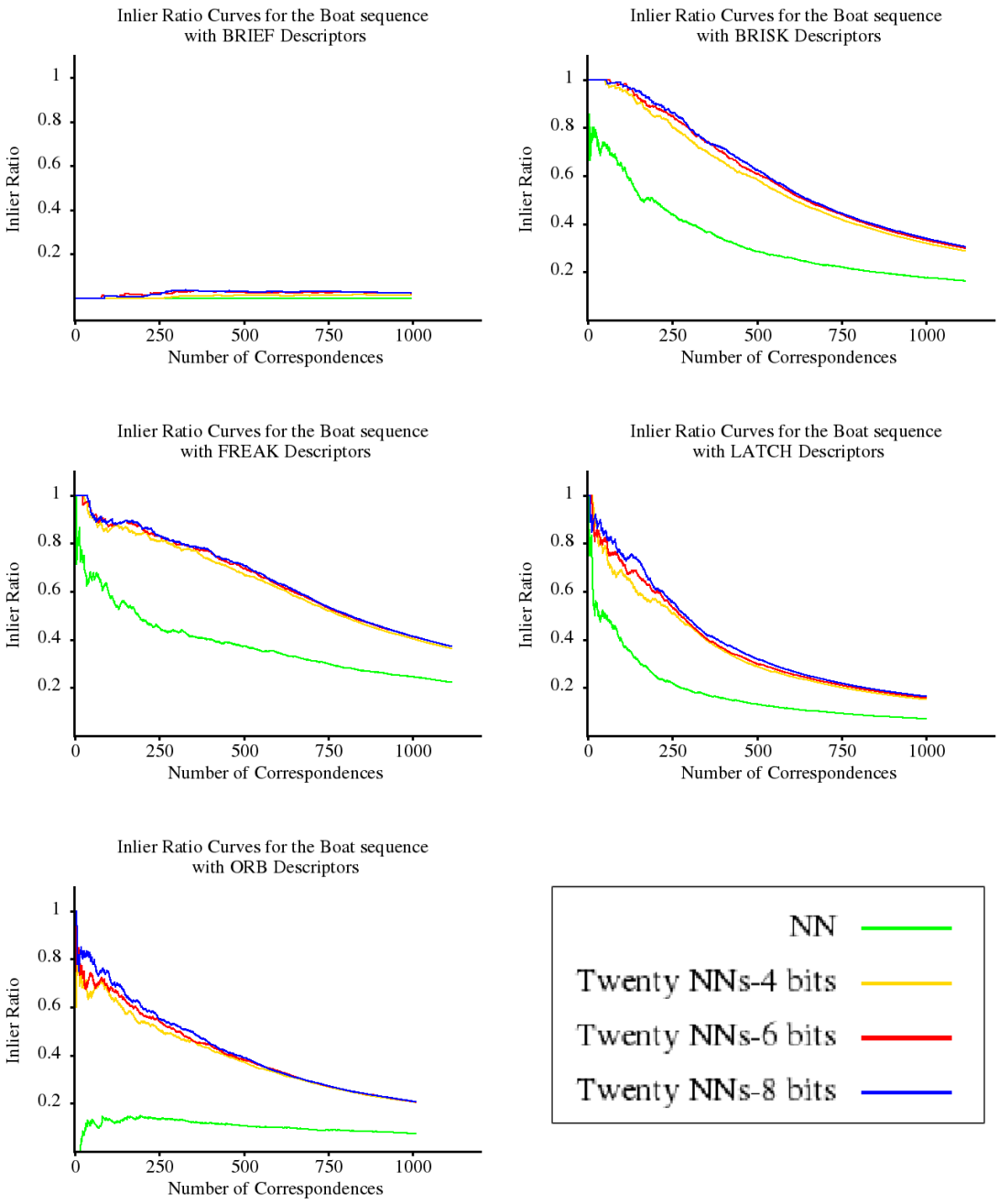


Figure B.14. Inlier ratio curves for the third test image of the Boat sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.

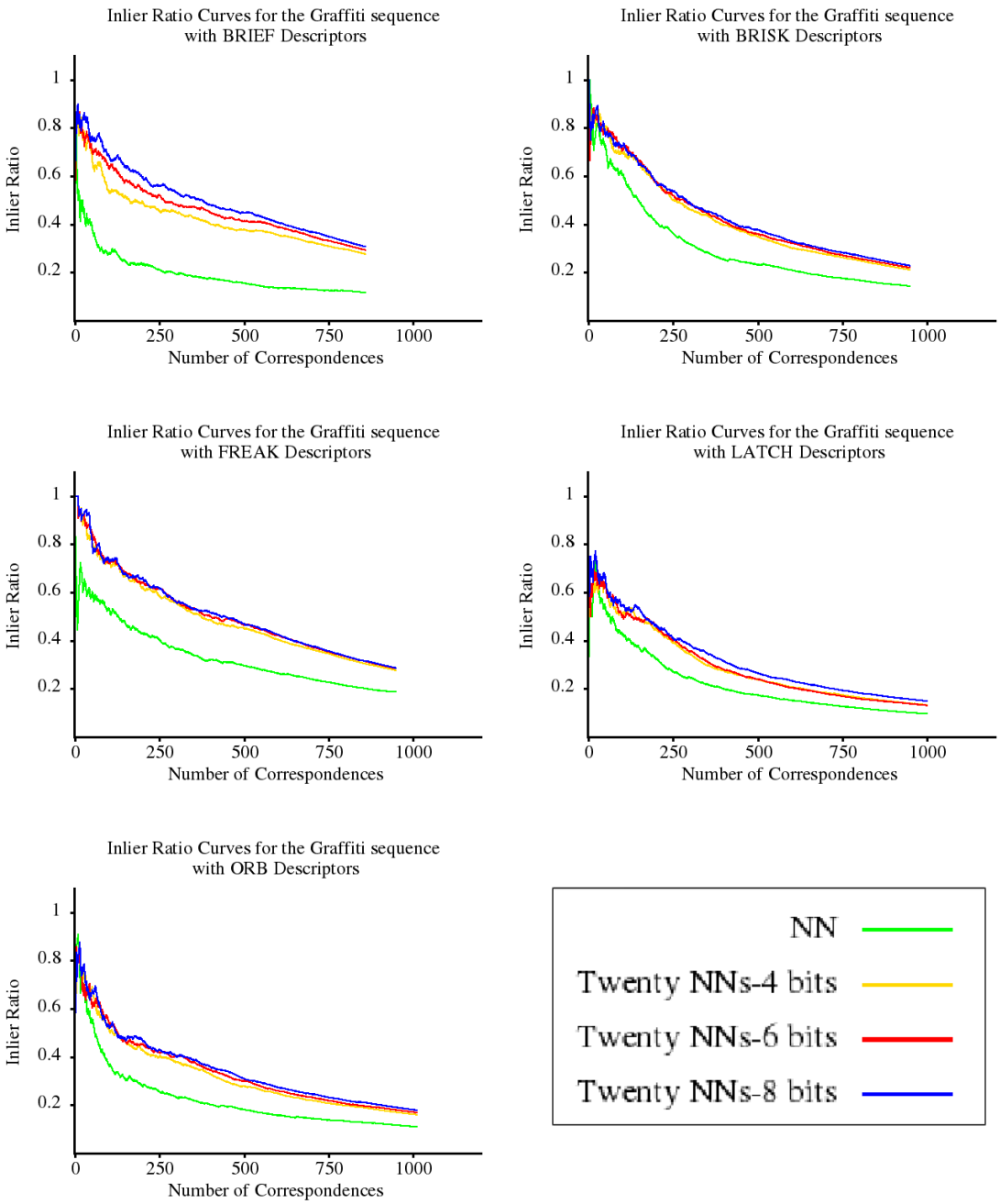


Figure B.15. Inlier ratio curves for the third test image of the Graffit sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.

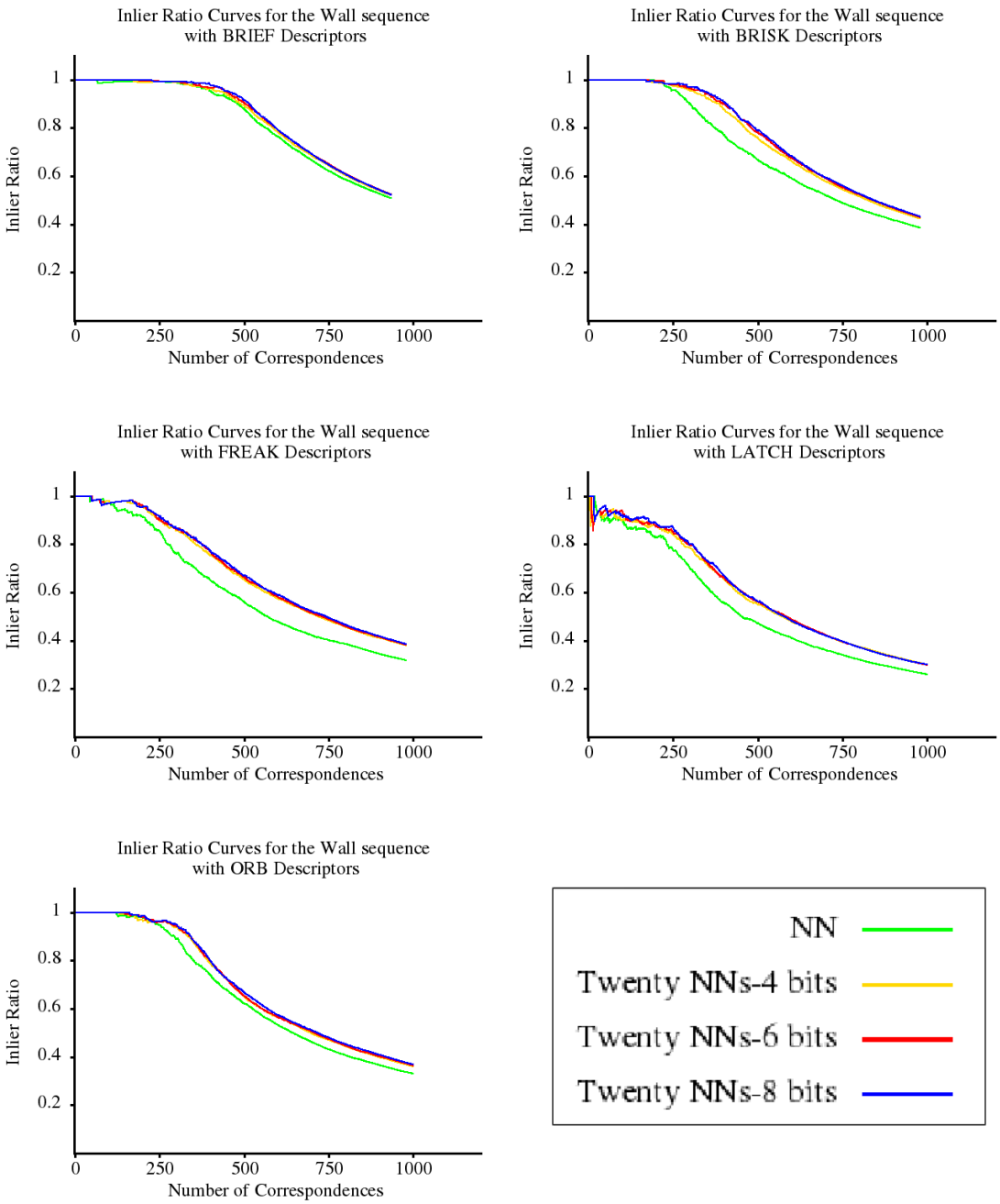


Figure B.16. Inlier ratio curves for the third test image of the Wall sequence with a single reference image and brute-force matches. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.

APPENDIX C

INLIER RATIO EXPERIMENTS BY USING LSH MATCHES

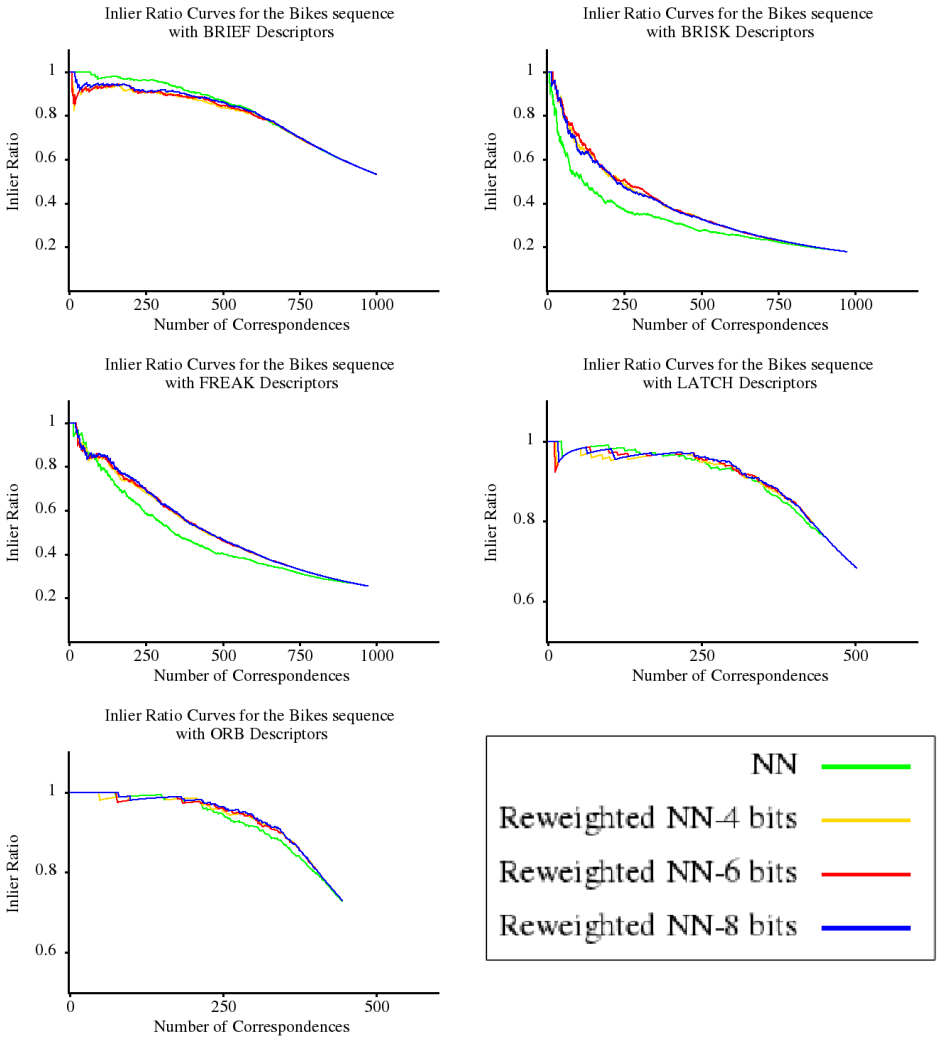


Figure C.1. Inlier ratio curves for the third test image of the Bikes sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

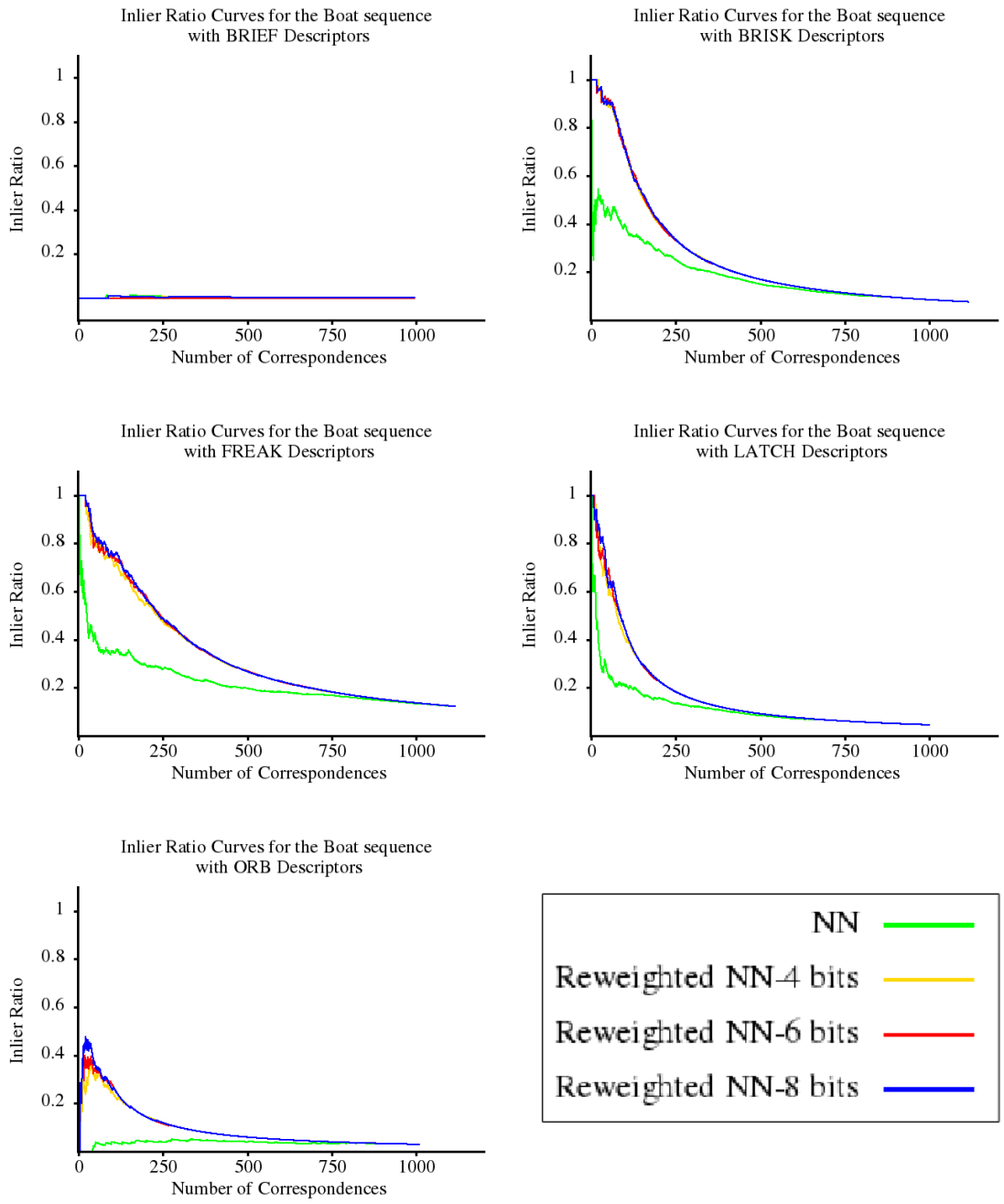


Figure C.2. Inlier ratio curves for the third test image of the Boat sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

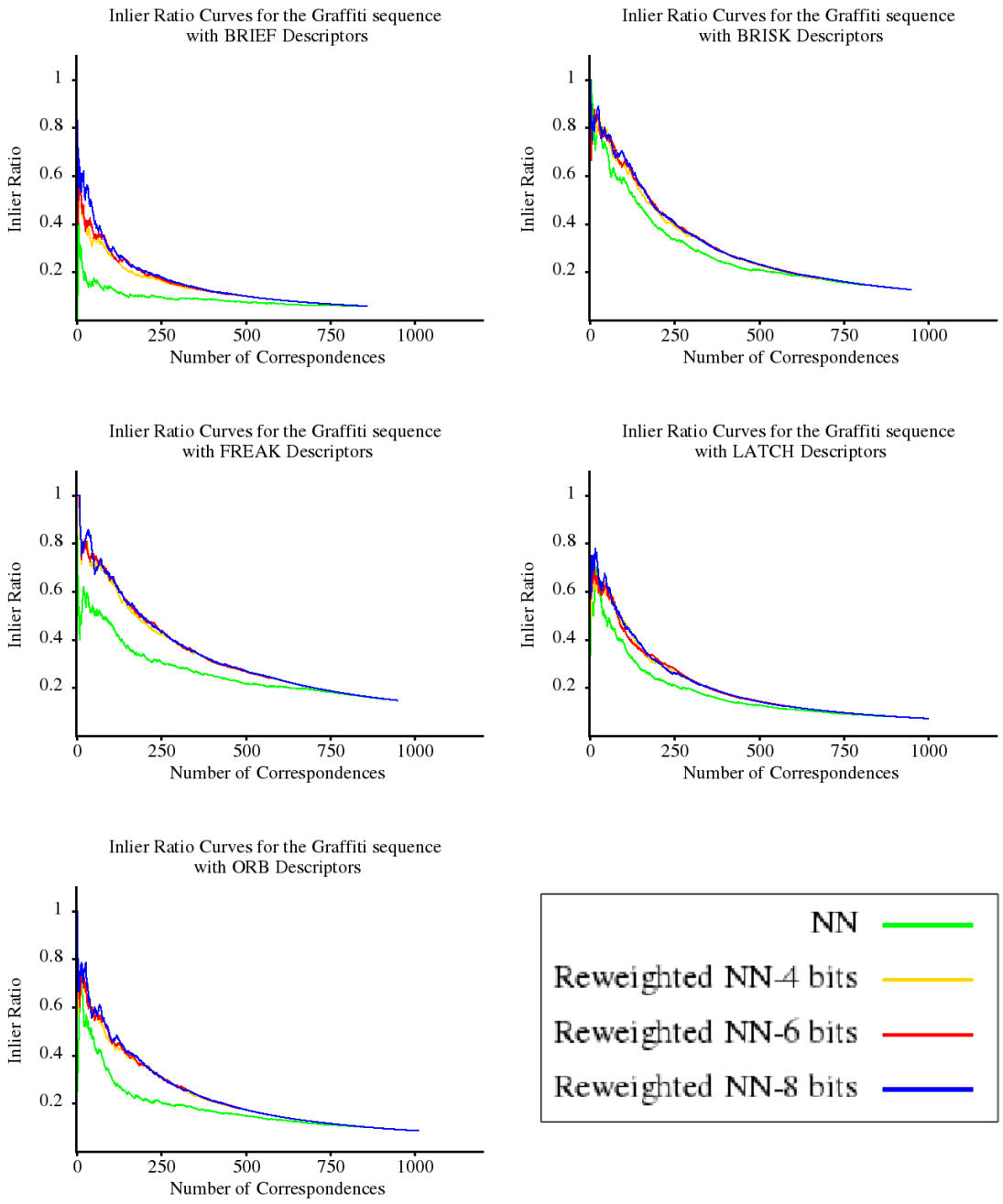


Figure C.3. Inlier ratio curves for the third test image of the Graffiti sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

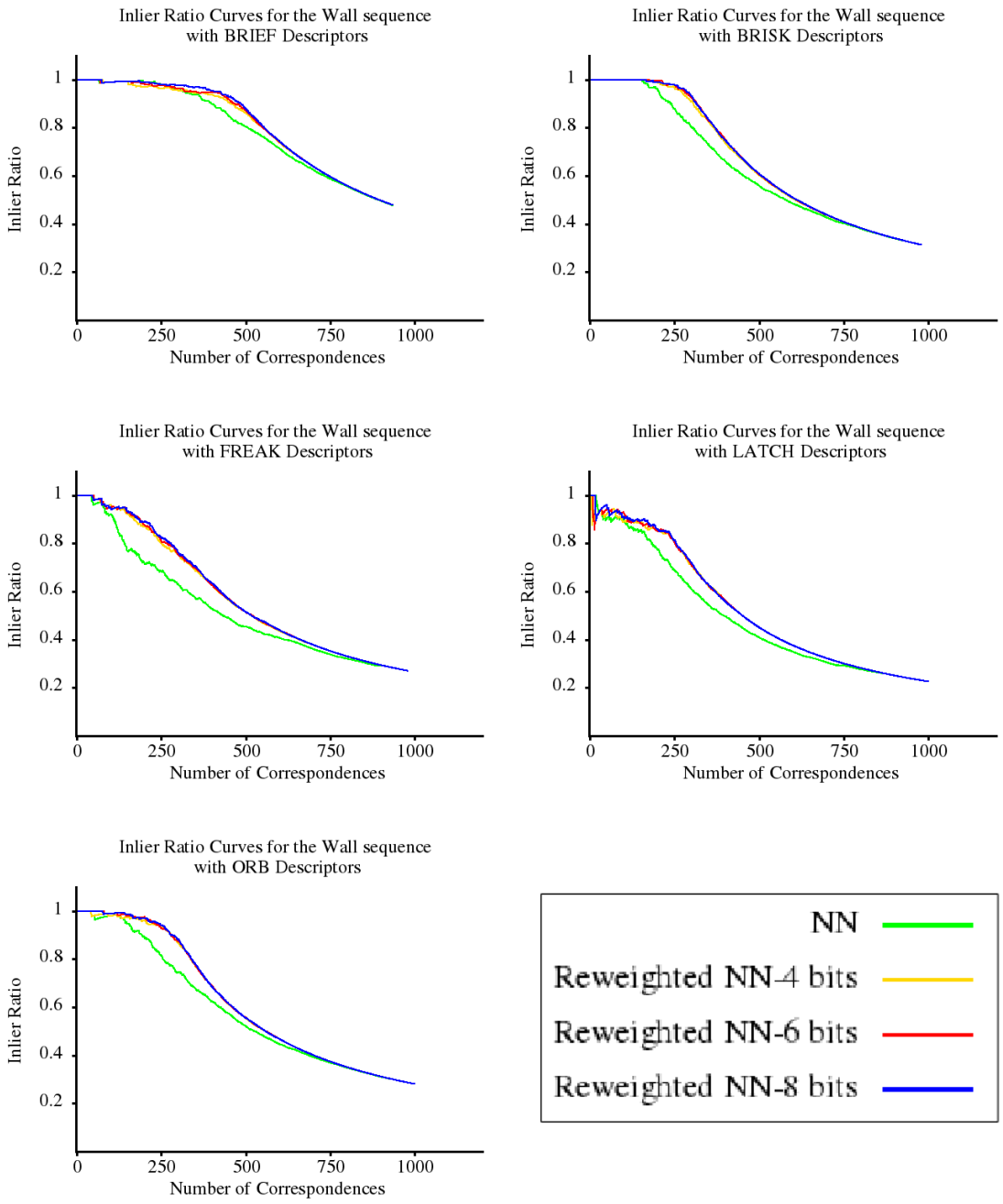


Figure C.4. Inlier ratio curves for the third test image of the Wall sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and by the keypoint specific score of Equation 5.3 (Reweighted NN) with bit groups of 4,6, and 8.

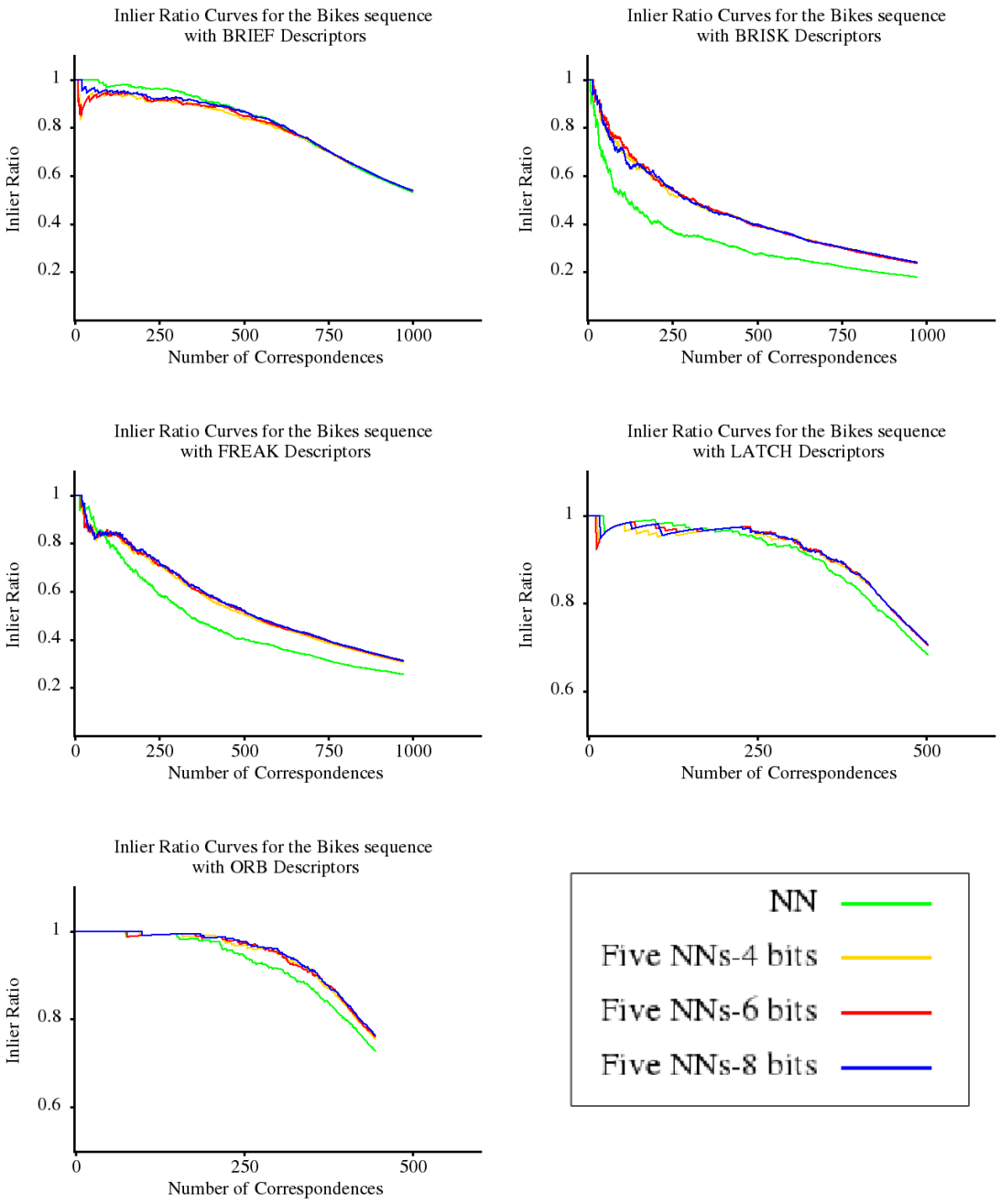


Figure C.5. Inlier ratio curves for the third test image of the Bikes sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

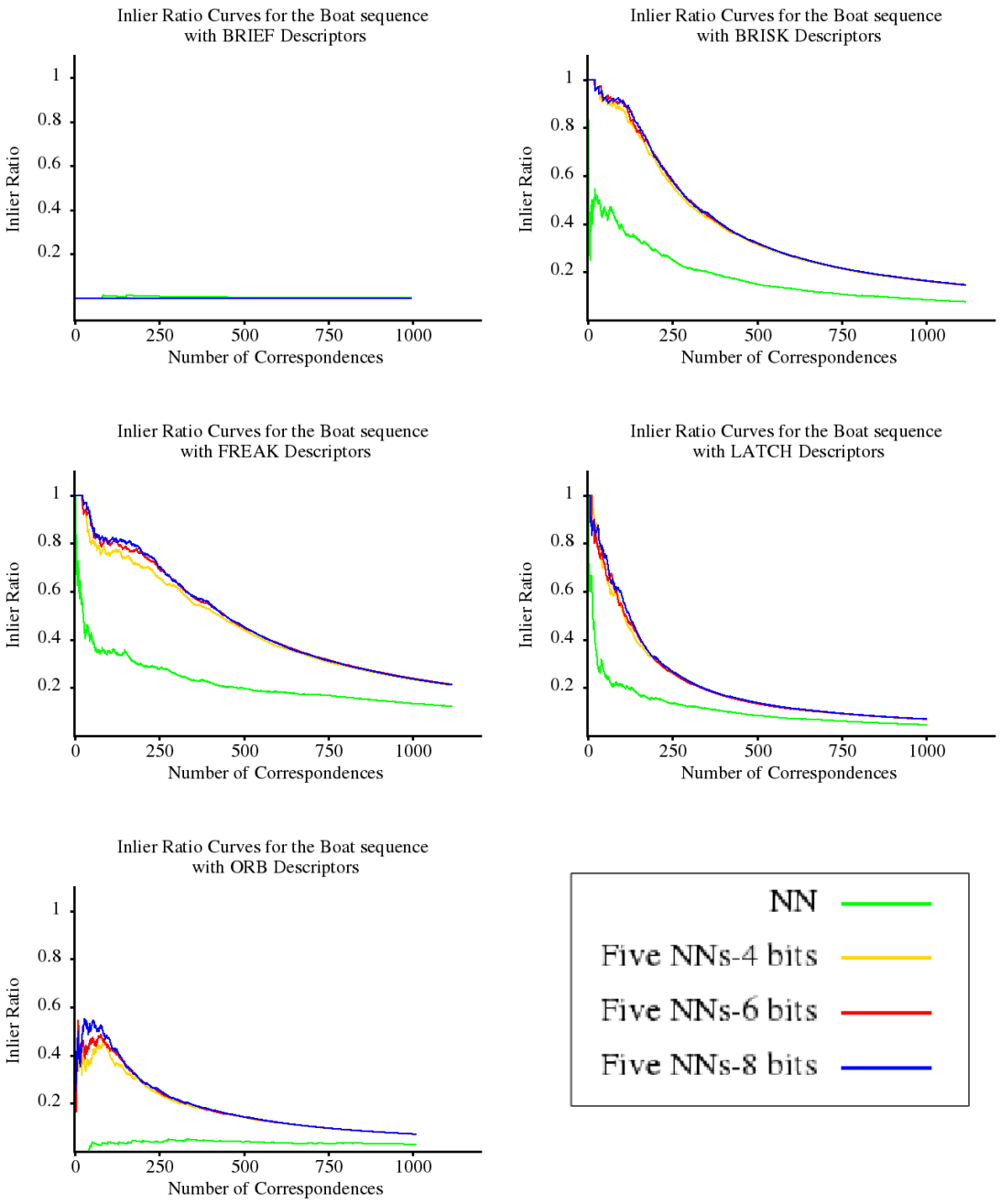


Figure C.6. Inlier ratio curves for the third test image of the Boat sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

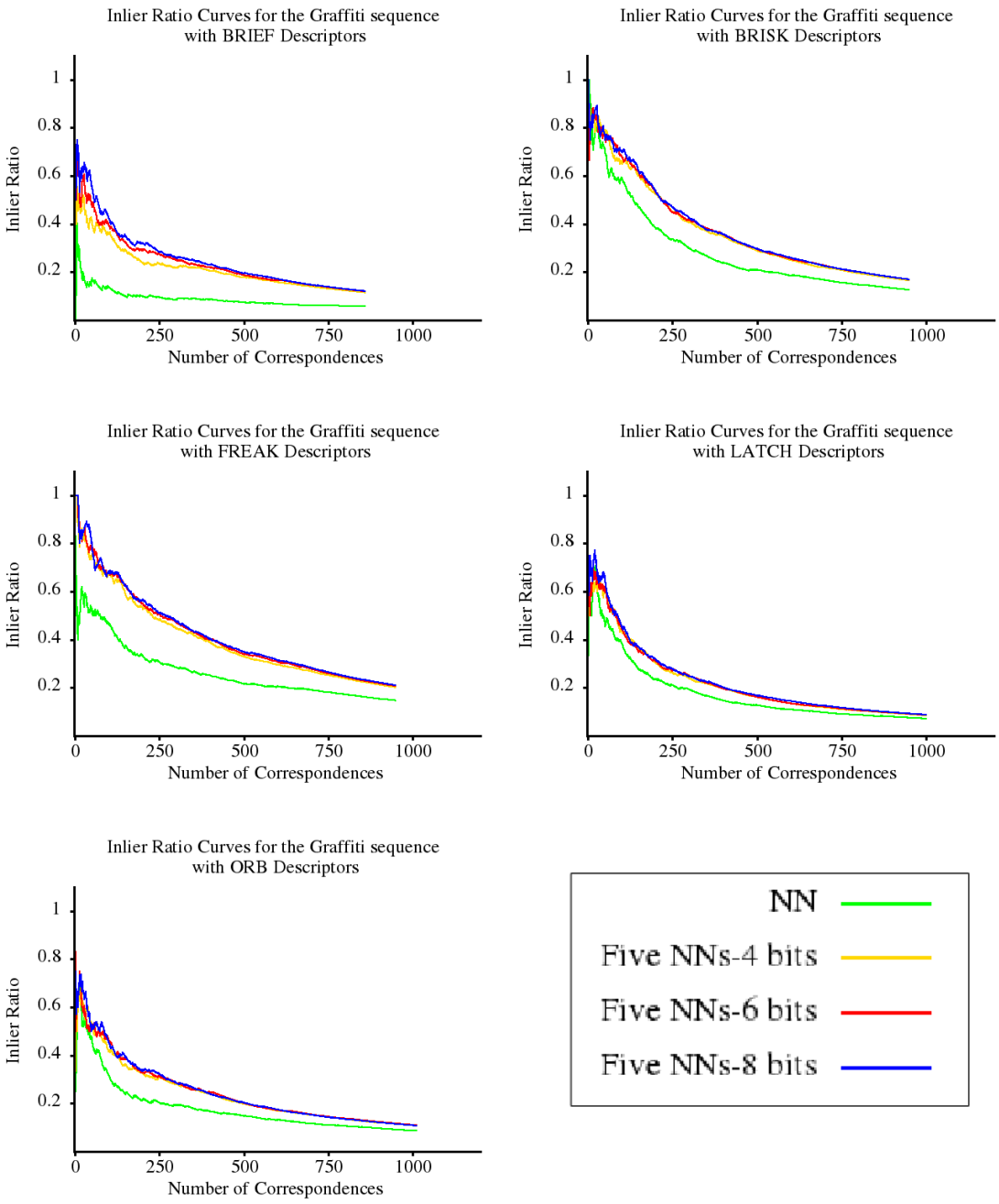


Figure C.7. Inlier ratio curves for the third test image of the Graffiti sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

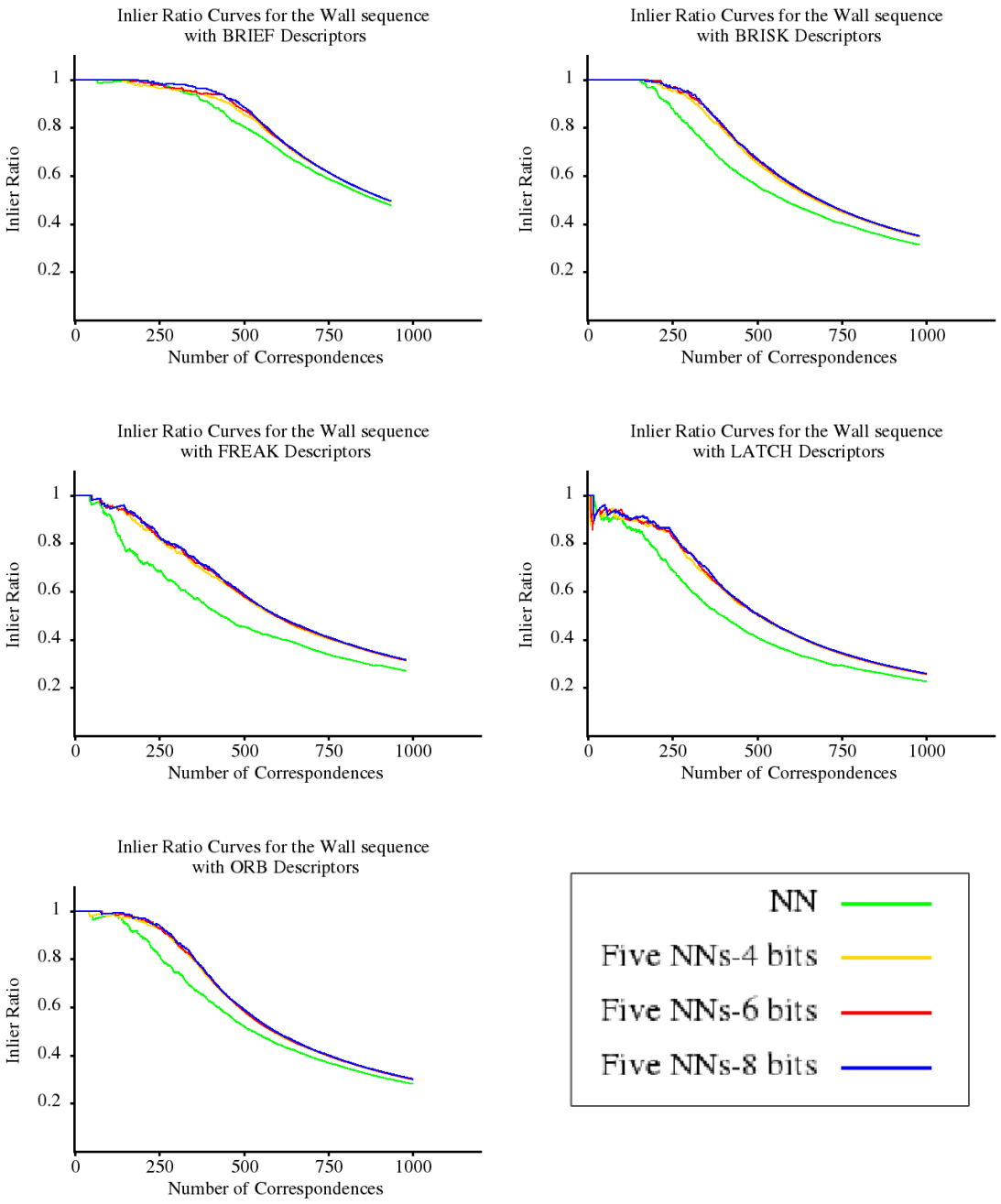


Figure C.8. Inlier ratio curves for the third test image of the Wall sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the five near neighbour list matches by Equation 5.3 (Five NNs) with bit groups of 4,6, and 8.

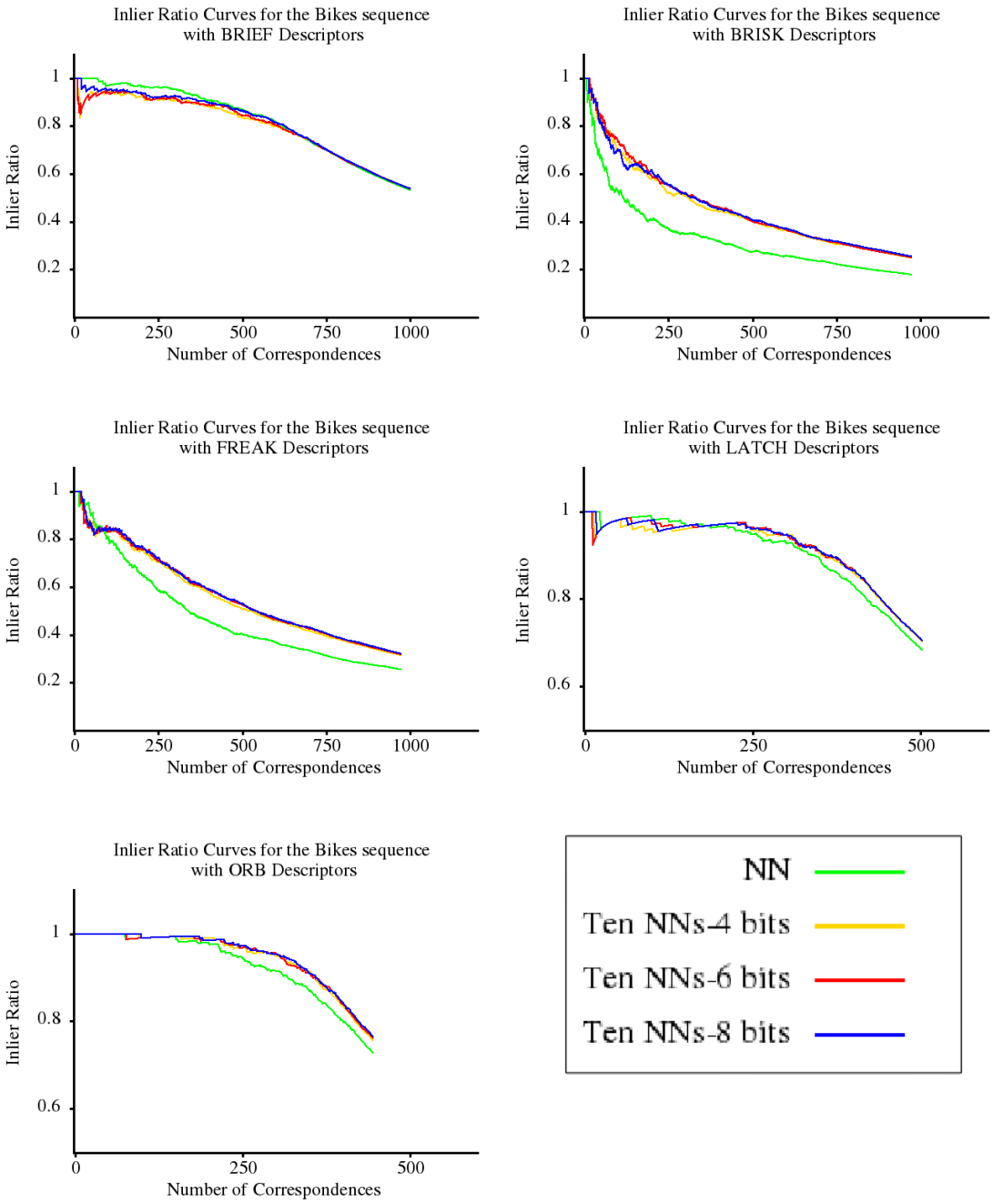


Figure C.9. Inlier ratio curves for the third test image of the Bikes sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

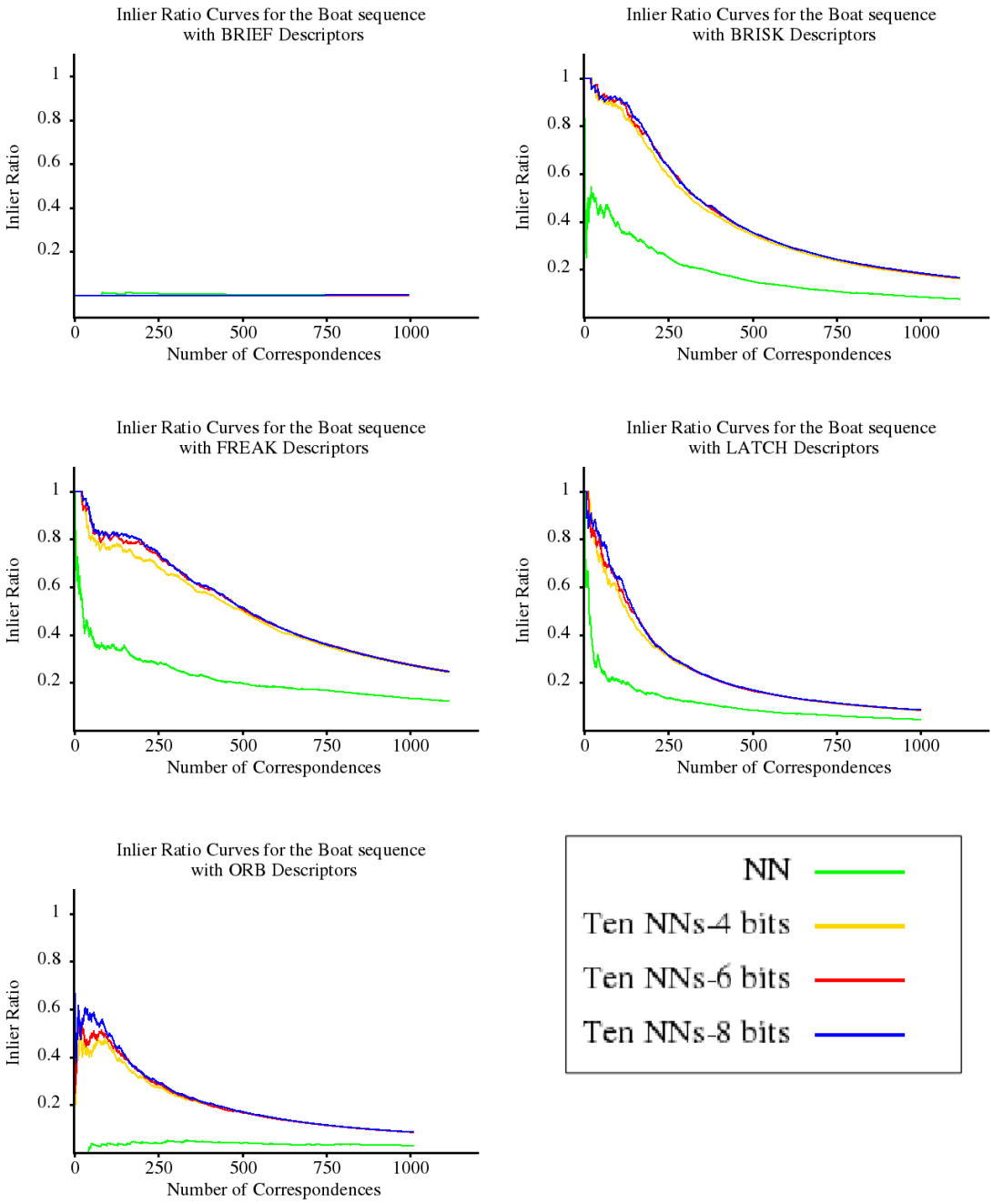


Figure C.10. Inlier ratio curves for the third test image of the Boat sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

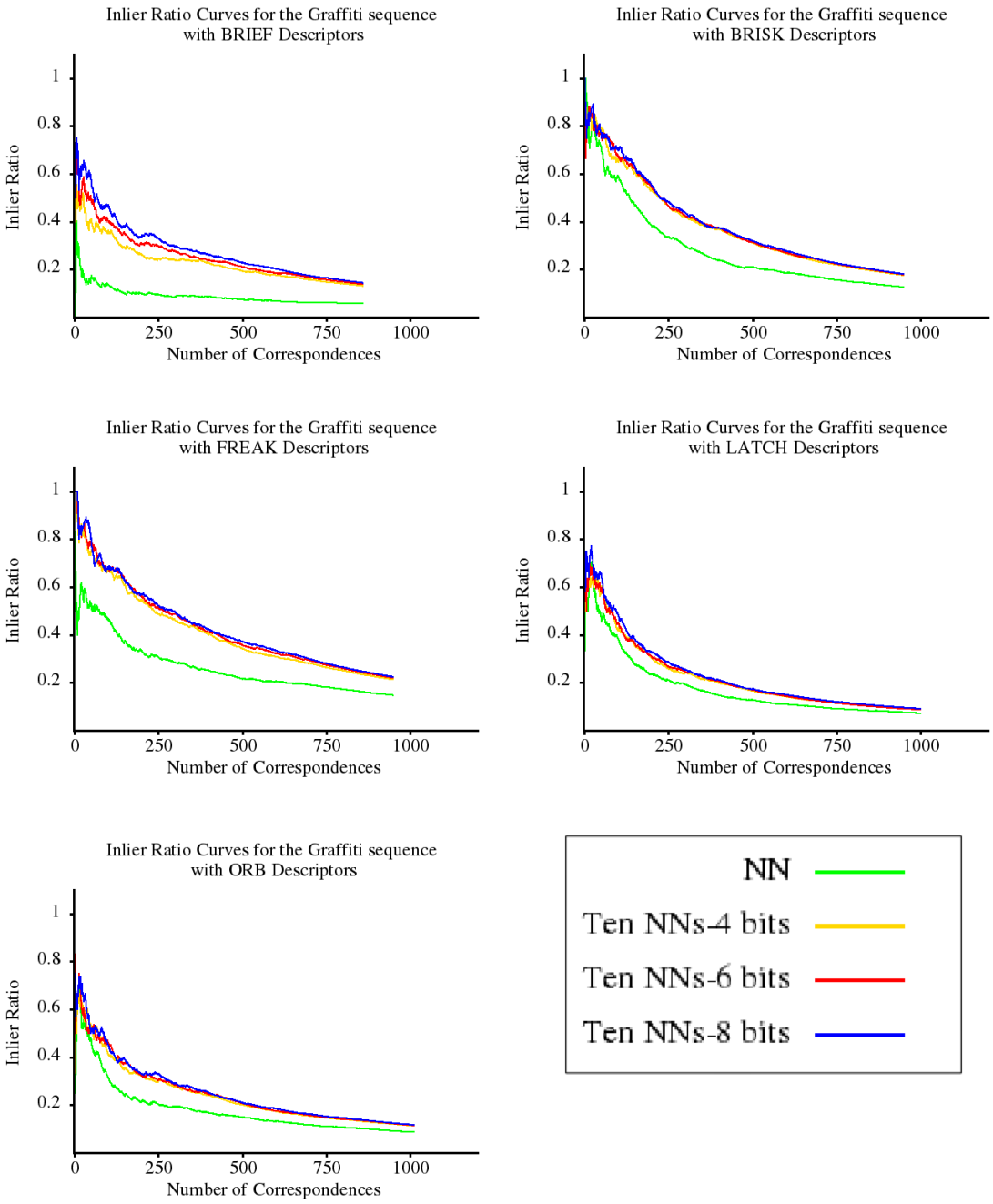


Figure C.11. Inlier ratio curves for the third test image of the Graffiti sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

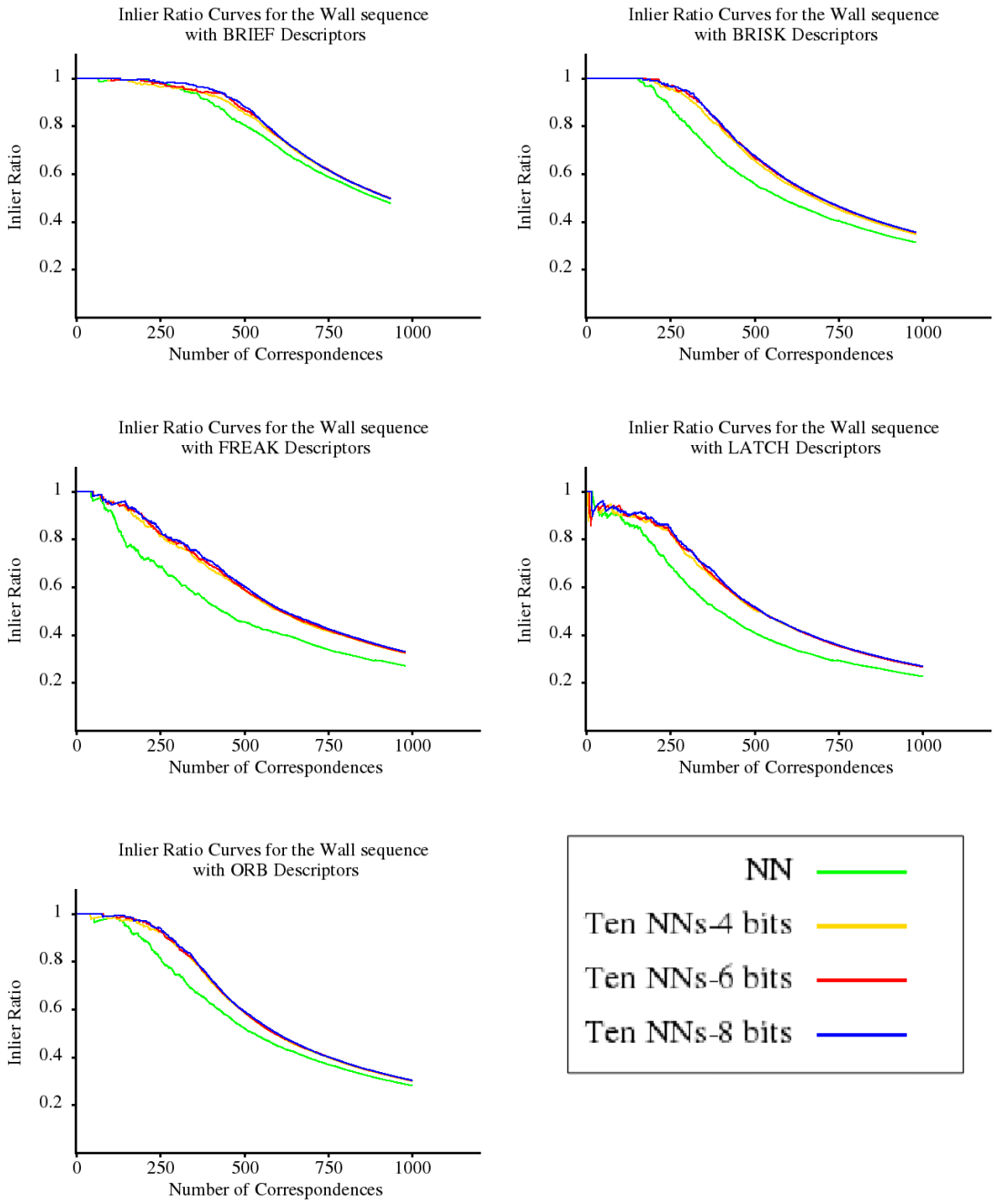


Figure C.12. Inlier ratio curves for the third test image of the Wall sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the ten near neighbour list matches by Equation 5.3 (Ten NNs) with bit groups of 4,6, and 8.

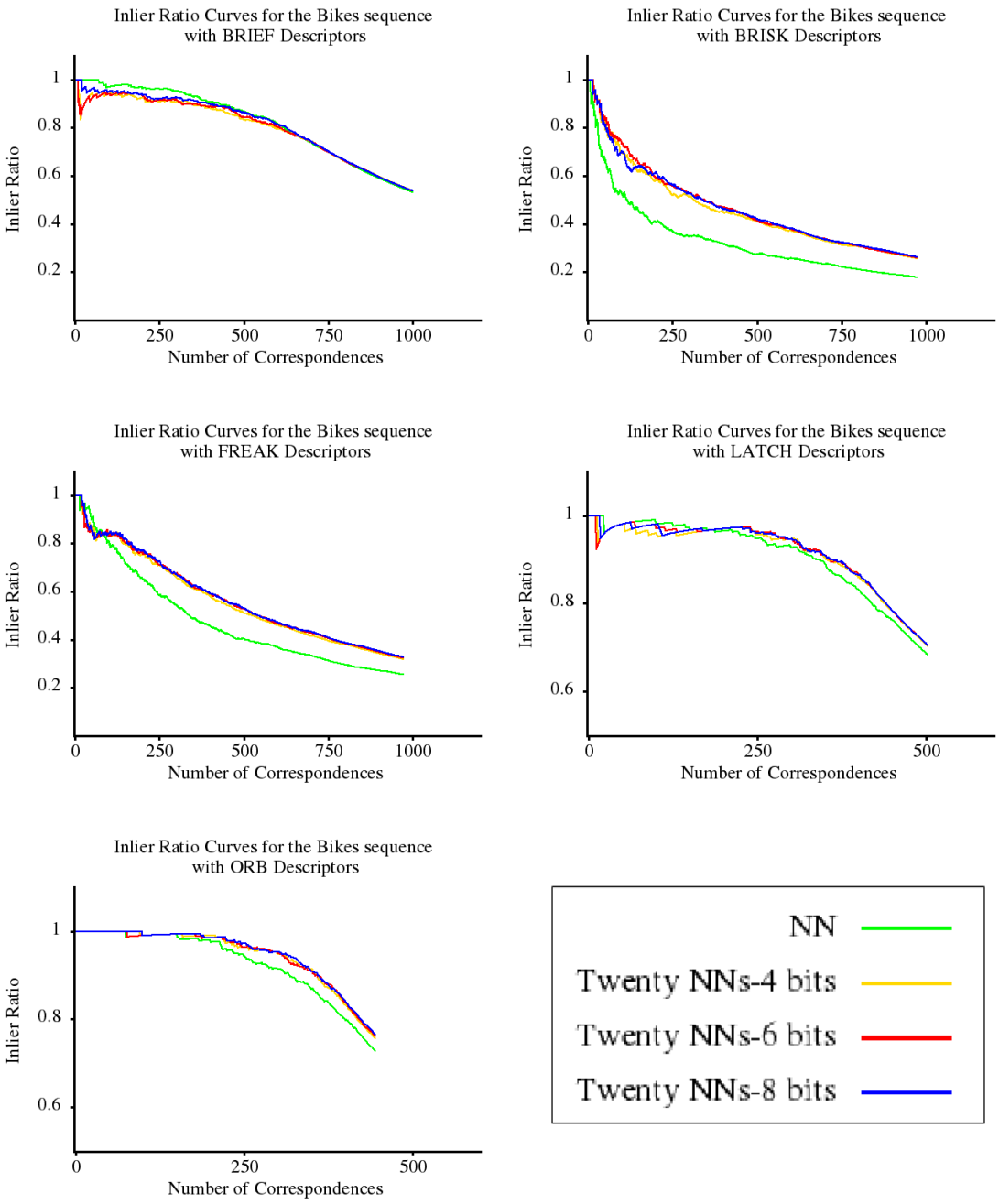


Figure C.13. Inlier ratio curves for the third test image of the Bikes sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.

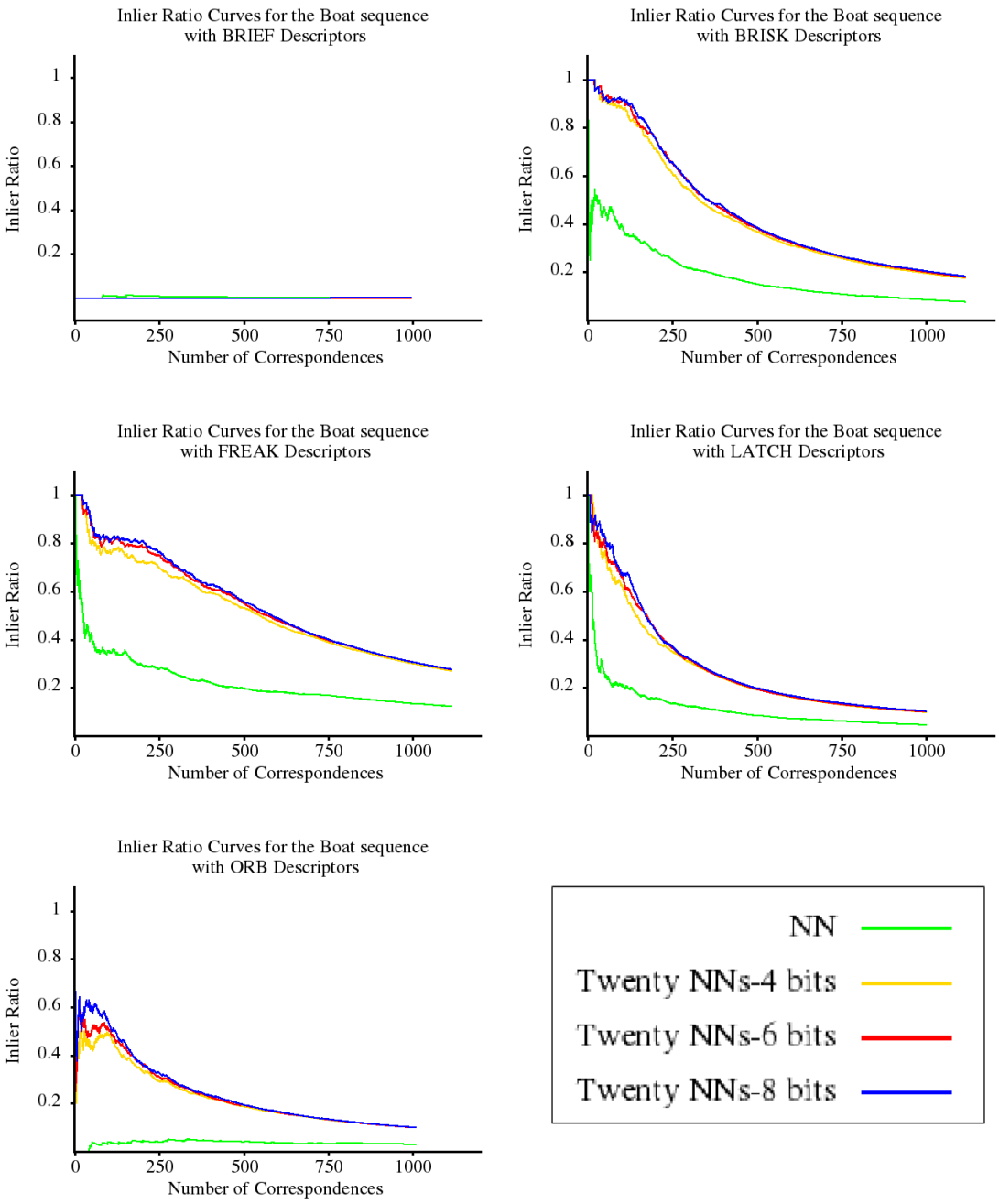


Figure C.14. Inlier ratio curves for the third test image of the Boat sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.

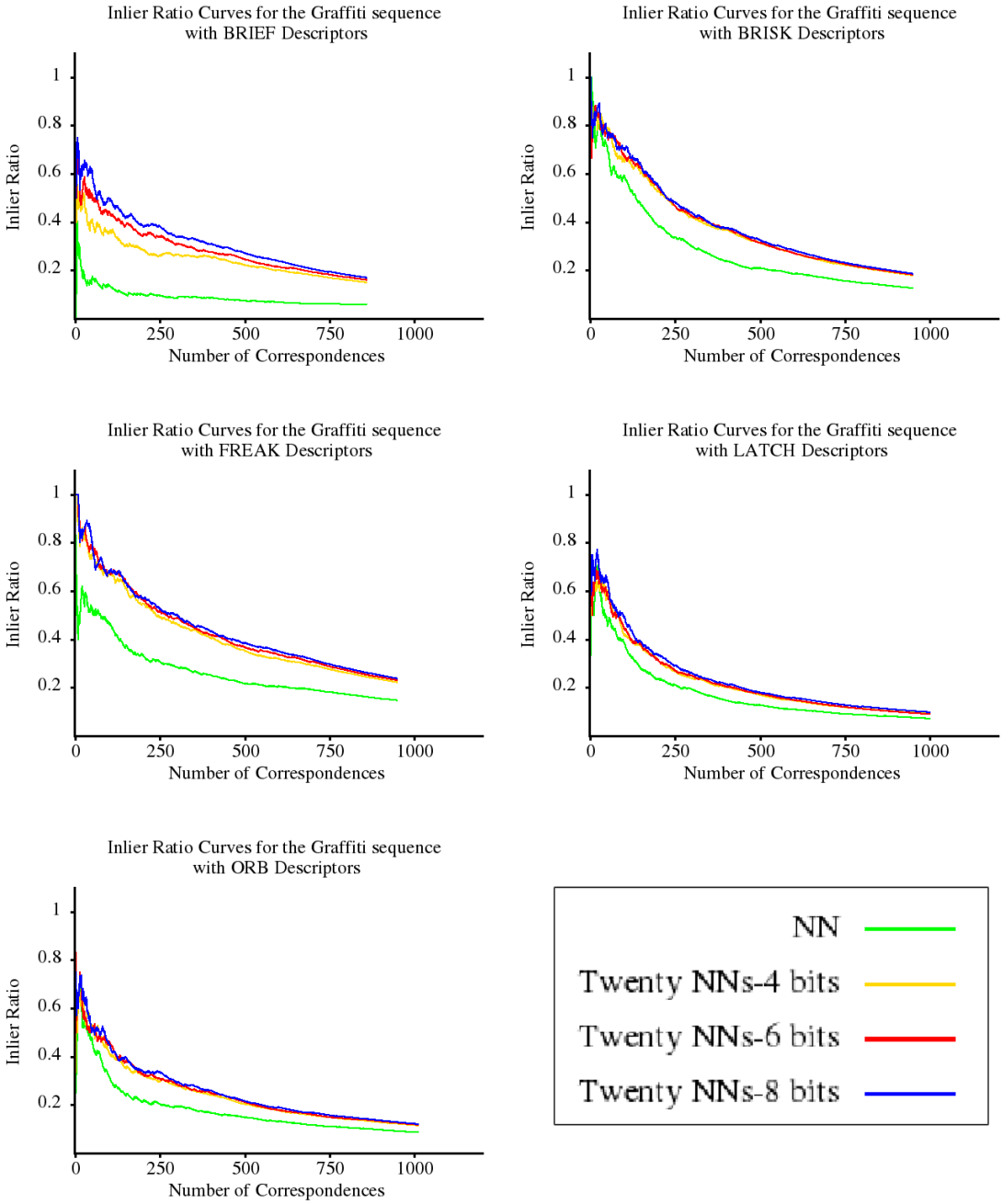


Figure C.15. Inlier ratio curves for the third test image of the Graffiti sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.

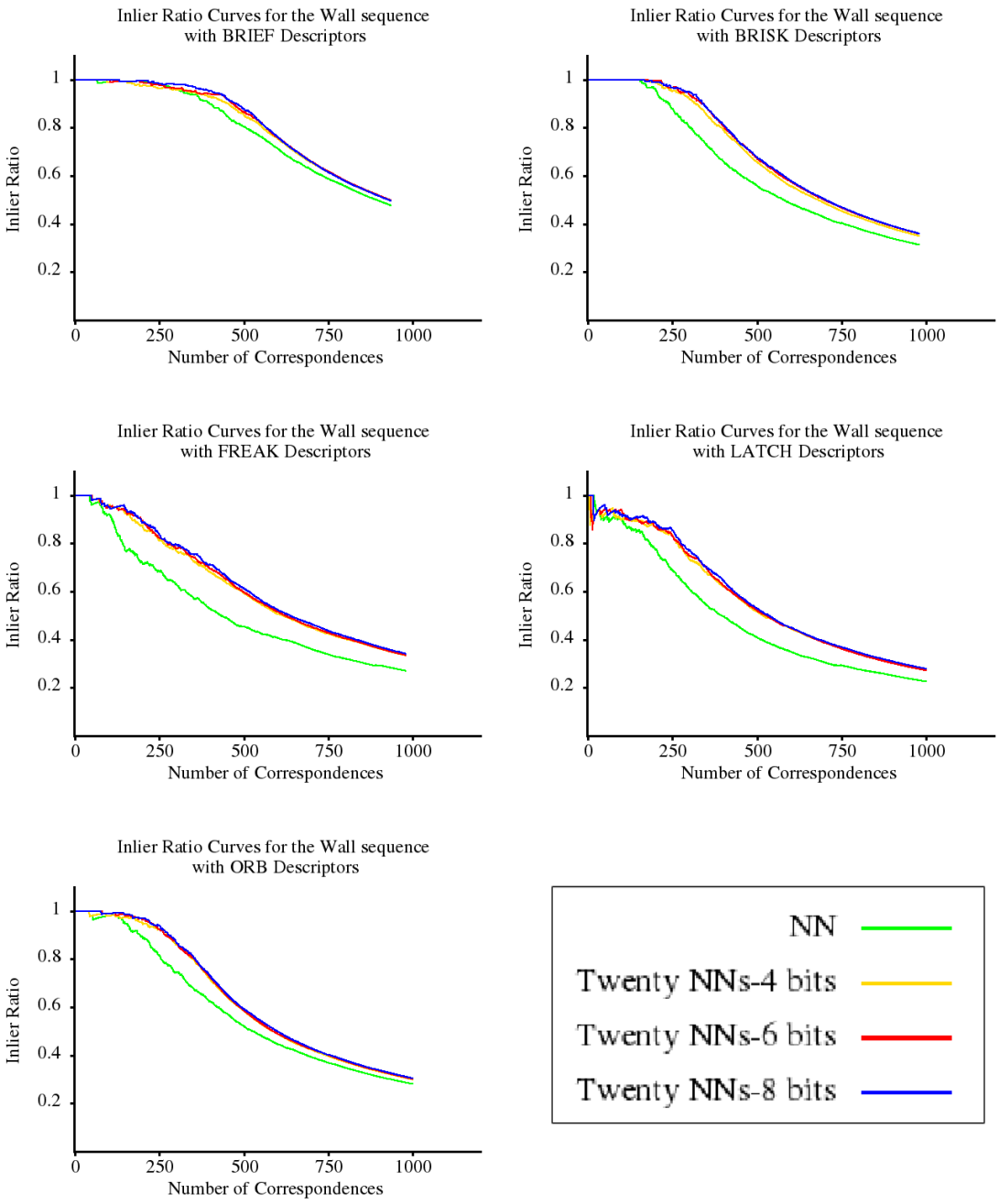


Figure C.16. Inlier ratio curves for the third test image of the Wall sequence when LSH is used to compute the near neighbour list over eight reference images. These curves obtained by ranking the nearest neighbour matches only by the descriptor distance (NN) and ranking the twenty near neighbour list matches by Equation 5.3 (Twenty NNs) with bit groups of 4,6, and 8.