LEARNING WEIGHTED HAMMING DISTANCE FOR BINARY DESCRIPTORS

Bin Fan[*] *Qingqun Kong*[*] *Xiaotong Yuan*[†] *Zhiheng Wang*[‡] *Chunhong Pan*[*]

[*] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[†] Rutgers University [‡] School of Computer Science and Technique, Henan Polytechnic University
bfan@nlpr.ia.ac.cn

ABSTRACT

Local image descriptors are one of the key components in many computer vision applications. Recently, binary descriptors have received increasing interest of the community for its efficiency and low memory cost. The similarity of binary descriptors is measured by Hamming distance which has equal emphasis on all elements of binary descriptors. This paper improves the performance of binary descriptors by learning a weighted Hamming distance for binary descriptors with larger weights assigned to more discriminative elements. What is more, the weighted Hamming distance can be computed as fast as the Hamming distance on the basis of a pre-computed look-up-table. Therefore, the proposed method improves the matching performance of binary descriptors without sacrificing matching speed. Experimental results on two popular binary descriptors (BRIEF [1] and FREAK [2]) validate the effectiveness of the proposed method.

Index Terms— Binary descriptor, Local descriptor, Image matching, Weighted Hamming distance

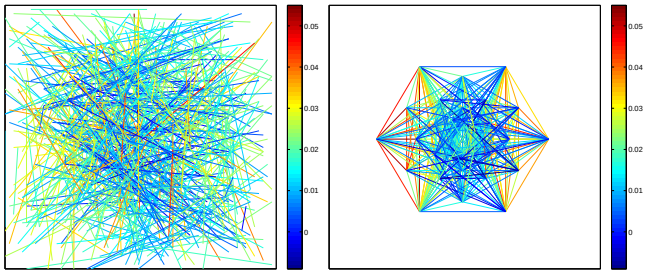
1. INTRODUCTION

With its success in many computer vision applications, local image descriptor has developed fast in the past decade. While the design of local descriptors is mainly focused on robustness in the past, the focus is moving to speed and storage with moderate robustness along with the increasing requirement for applying computer vision algorithms on embedded devices, such as mobile phones. However, traditional local descriptors are usually with large memory cost thus can not be extracted and matched in real time, such as SIFT [3], MROGH [4], LIOP [5], KAZE [6] to name a few. Therefore, these descriptors can not be used in mobile applications despite of their robustness. In order to fulfill the requirements of low memory cost and computational burden encountered in embedded applications, binary descriptors are proposed recently.

BRIEF (Binary Robust Independent Elementary Feature) [1] is a pioneer of binary descriptors. It uses intensity tests of randomly selected point pairs to construct local descriptor for an image patch. BRIEF is fast to compute since it only needs 512 comparisons for a 512 dimensional BRIEF descriptor. Due to its binary property, BRIEF is with low memory cost and can be fast matched by Hamming distance. It has been applied successfully in SLAM with small rotation and scale changes. One disadvantage of BRIEF is that it is sensitive to in-plane rotation and scale changes. This disadvantage severely limits its applicability. To alleviate this problem, Rublee et al. [7] proposed ORB (Oriented Fast and Rotated BRIEF), which detects FAST corners [8] in several octaves in order to achieve scale invariance approximately and selects robust binary tests for rotated patch. Therefore, ORB is claimed to be robust to rotation and scale changes. BRISK (Binary Robust Invariant Scalable Keypoints) [9] is proposed for the same purpose as ORB by applying a scale-space FAST-based detector [10], and a carefully designed sampling patterns for brightness comparisons to construct binary descriptor as well as for orientation estimation. In BRISK, point pairs are generated by the predefined sampling patterns and they are divided into short-distance and long-distance subsets. The long-distance subset is used to estimate orientation so as to obtain rotation invariance, while the short-distance subset is used to construct BRISK descriptor on the rotated image patch. Inspired by the human visual system, Alahi et al. [2] proposed the FREAK (Fast Retina Keypoint) descriptor.

While much work is focused on designing binary descriptors, rare work is conducted on matching binary descriptors. A key advantage of binary descriptors is that the Hamming distance can replace the widely used Euclidean distance when matching descriptors. However, it imposes equal emphases on all elements of a binary descriptor by using the Hamming distance. Unfortunately, it is less likely that all elements of a binary descriptor contribute equally to the robustness and distinctiveness of this descriptor.

Contributions: In this paper, we propose to use a weighted Hamming distance for visual matching. By assigning larger weights to more important elements, the weighted Hamming distance is expected to better separate matching image patches from non-matching image patches. To this end, weights are learnt from training samples by minimizing a margin-based empirical loss. Meanwhile, in order to compute the weighted Hamming distance efficiently, a look-up-



(a) (b)

Fig. 1. The learnt weights for (a) BRIEF and (b) FREAK. The box represents an image patch. Each line in the box links a pair of points used for intensity test with its color indicating weight.

table based strategy is proposed. With such a strategy, the weighted Hamming distance can be computed as fast as Hamming distance with only several k-bytes memories additional for storing look-up-tables. To show the effectiveness of the proposed method, we have learnt the weighted Hamming distance for two popular binary descriptors, BRIEF [1] and FREAK [2], respectively. The learnt weights of them are shown in Fig. 1. Experimental evaluation on the Patch Dataset [11] demonstrates that the performance of binary descriptors is significantly improved by the weighted Hamming distance.

The rest of this paper is organized as follows. Section 2 describes the objective for learning weighted Hamming distance. Then, a fast method to compute the weighted Hamming distance is elaborated in Section 3. Experiments are reported in Section 4, followed by a brief introduction of the related work in Section 5. Finally, we conclude this paper in Section 6.

2. LEARNING WEIGHTED HAMMING DISTANCE

Given two image patches x and y, denote their binary descriptors as $b(x) \in \{0,1\}^n$ and $b(y) \in \{0,1\}^n$ respectively. Then their Hamming distance is computed by:

$$\text{Ham}(x,y) = \sum_{i=1}^{n} b_i(x) \otimes b_i(y) \tag{1}$$

in which n is the dimension of binary descriptor and $\otimes$ stands for bitwise XOR operation.

According to the definition of Hamming distance, it is clear that all the elements of a binary descriptor contribute equally to the distance. However, the distinctiveness of each element is usually different. Therefore, assigning equal weights to all elements may degrade the performance of binary descriptors. A more reasonable way is to learn different weights for different elements of binary descriptors. Here, we generalize the Hamming distance to the weighted Hamming distance, defined as:

$$\text{WHam}(x,y) = \sum_{i=1}^{n} w_i(b_i(x) \otimes b_i(y)) \tag{2}$$

where w_i is the weight of the ith element. The goal of this paper is to learn $w_i, i = 1, 2, \cdots, n$ for a given type of binary descriptor (e.g. BRIEF) based on a set of training samples.

By assigning different weights to binary codes, what we expect is to obtain a distance space in which the distances of matching patches are less than those of non-matching patches. It is noted that in the domain of wide-baseline image matching, the absolute value of feature descriptors' distance is less significant than the relative ranking. This is because that feature matching is usually obtained by searching the nearest neighbor in the descriptor space. In this case, distance ranking is more important. Therefore, our objective for learning the optimal weights in Equ. (2) is a margin-based one. Specifically, we impose the following constraint on the weighted Hamming distances of training samples:

$$\text{WHam}(X_i) < \text{WHam}(X_j) - 1, \forall (X_i \in \mathcal{P}, X_j \in \mathcal{N}) \tag{3}$$

in which $\text{WHam}(X_i)$ represents $\text{WHam}(x_i, y_i)$, while $\mathcal{P}$ and $\mathcal{N}$ denote the training sets of matching pairs and non-matching pairs respectively. The constraint presented in (3) means that in the weighted Hamming space, the distances of matching pairs are less than those of non-matching pairs by at least one unit. Consequently, we obtain the empirical loss on a given training set:

$$loss(w) = \sum_{X_i \in \mathcal{P}} \sum_{X_j \in \mathcal{N}} \max\{\text{WHam}(X_i) - \text{WHam}(X_j) + 1, 0\}$$

Therefore, our optimization problem can be written as:

$$w = \arg\min_{w} (loss(w) + \lambda R(w)) \tag{4}$$

in which $R(w)$ is a regularized term and λ is a parameter trading-off between the empirical loss and regularization. We use $L2$ norm for regularization, i.e., $R(w) = \|w\|_2^2$ and set λ to be 1 in our experiments. Since our objective function is differentiable, the optimization problem presented in Equ. (4) can be easily solve by the gradient descent algorithm.

3. FAST COMPUTATION OF THE WEIGHTED HAMMING DISTANCE

The most attractive property of Hamming distance is that it can be fast computed in modern computers by bitwise XOR operations. For the weighted Hamming distance, it will take much more time to compute (cf. Table 1). Therefore, we propose a strategy for fast computation of the weighted Hamming distance by look-up-tables. It can be computed as fast as Hamming distance, with only additional several addressing operations.

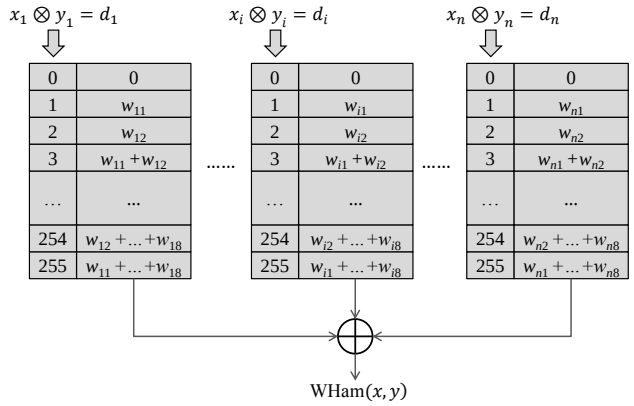


Fig. 2. Fast computation of the weighted Hamming distance with look-up-tables.

The computation procedure of the weighted Hamming distance is shown in Fig. 2. Firstly, the input binary descriptors x and y are separated into several bytes (if their dimensionality can not be divided by 8, fill zeros in high positions.), denoted as $(x_1, x_2, \cdots, x_n)$ and $(y_1, y_2, \cdots, y_n)$; secondly, for each separated byte x_i and y_i, bitwise XOR is conducted to obtain a byte-type value d_i; thirdly, for a byte-type value $d_i \in [0, 255]$, accessing the ith look-up-table with the address d_i and getting a float value v_i; finally, summing over all the obtained float values as the weighted Hamming distance of x and y, i.e., WHam$(x, y) = v_1 + v_2 + \cdots + v_n$. Therefore, by using look-up-tables, the computation of the weighted Hamming distance only requires additional $n/8$ addressing operations than the computation of Hamming distance.

From the above description of the look-up-table based computation, $n/8$ look-up-tables are required, each of which has $2^8 = 256$ elements. They are constructed by summing the corresponding weights. Take the 5th element (its address is 0x00000101) in the ith table for example, its value is $w_{i1} + w_{i3}$ and $w_{ij} = w_{8\times(i-1)+j}$. Note that for each look-up-table, only $256 \times 4 = 1024$ bytes storage is required. Consequently, comparing to Hamming distance, there are only n/8 k-bytes additional memories required in computing the weighted Hamming distance.

4. EXPERIMENTS

In order to show the effectiveness of the weighted Hamming distance for binary descriptors, we have conducted experiments on the Patch Dataset [11]. Two popular binary descriptors are used in our experiments: BRIEF [1] and FREAK [2]. BRIEF is the pioneer binary descriptor in the literature used for real-time image matching while FREAK is the most recent one. We used the implementations provided on the authors' websites.

The Patch Dataset contains up to 500k image patch pairs for each of the three scenes, namely Liberty, Notre Dame and Yosemite. For each scene, there are equal numbers of matching pairs and non-matching pairs. These patch pairs were sampled around interest points and the correspondences between patches were found by mapping between images using stereo depth maps obtained by the Photo Tourism algorithm [12] and Michael Goesele's multi-view stereo algorithm [13]. These patches are scale and rotation normalized and of size 64×64.

4.1. Performance Evaluation

To show the effectiveness of our method, we learnt the weighted Hamming distance on Liberty, Notre Dame and Yosemite subsets respectively, and then tested its performance on the other subsets. Therefore, there are totally 6 train/test configurations in our evaluation. We used 10k samples for training and 100k samples for testing with equal number of matching and non-matching pairs in each configuration. Results are reported in terms of ROC curves which are obtained by varying a threshold on the distances between patch pairs in the descriptor space. Fig. 3 shows the experimental results. It is clear to see that the performance of BRIEF and FREAK is significantly improved with the weighted Hamming distance. Since the point pairs used for intensity comparisons in FREAK have been optimized while BRIEF uses randomly selected point pairs, the improvement obtained with FREAK is not as significant as that obtained with BRIEF.

4.2. Timing tests

In Section 3, we have theoretically analyzed complexity of the proposed strategy for fast computation of the weighted Hamming distance. To validate its efficiency, we have conducted speed tests on a laptop with an Intel Core 3.1GHz CPU and 4GB memories. For BRIEF and FREAK, we calculated Hamming distance and the weighted Hamming distance on 100,000 patch pairs and recorded the corresponding times respectively. The tests are repeated 1000 times with the average results listed in Table 1. As expected, the matching times for BRIEF and FREAK are identical[1] since they have the same dimensions. The computation time of the weighted Hamming distance is slightly longer than that of Hamming distance due to the additional address accessing operations for computing the weighted Hamming distance. Table 1 also tabulates the computation time of the weighted Hamming distance without the proposed strategy. It is nearly 20 times longer than that using the proposed strategy, demonstrating the efficiency of the proposed strategy for calculating the weighted Hamming distance.

[1]The slightly difference between the timing results of BRIEF and FREAK is due to system error.

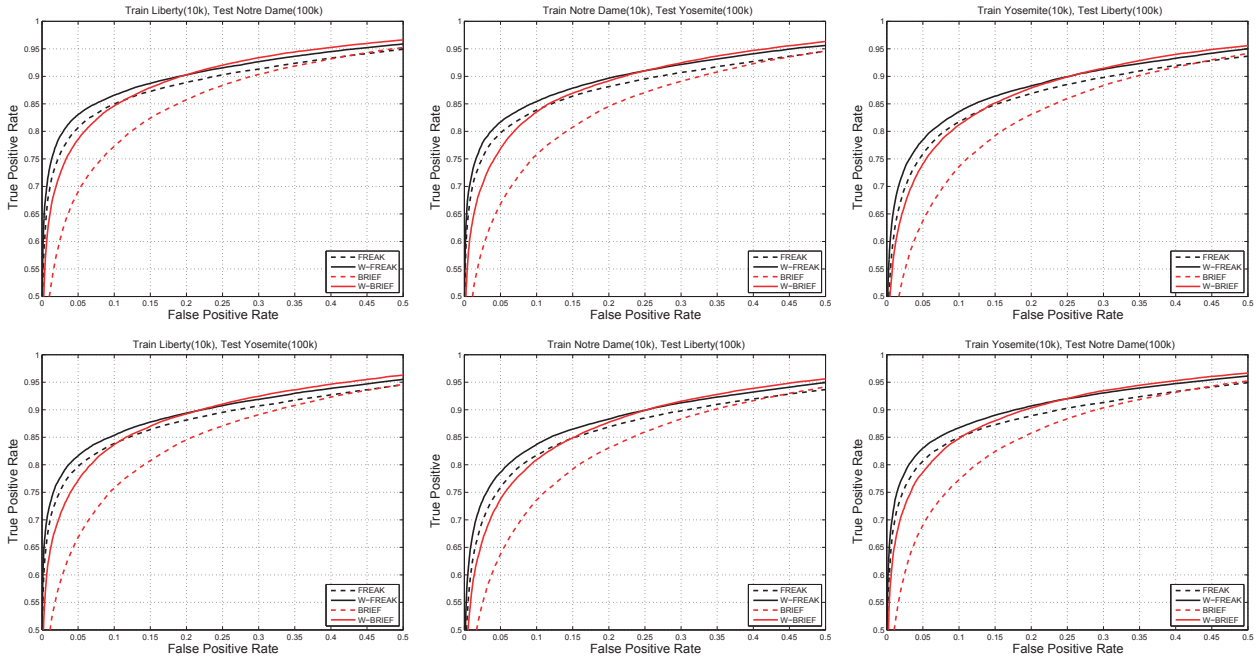


Fig. 3. Experimental results across all splits of training and testing sets. W-BRIEF and W-FREAK denote results obtained in the weighted Hamming distance learnt by BRIEF and FREAK respectively. There are 10k samples used for training and 100k samples used for testing.

Table 1. Average matching times (ns) for BRIEF and FREAK with Hamming distance (Ham) and the weighted Hamming distance (WHam) respectively. For the purpose of comparison, the time for directly calculating the weighted Hamming distances is also included. LUT stands for the proposed look-up-table strategy.

BRIEF	Ham	WHam (LUT)	WHam
	49.3	56.5	970.8
FREAK	Ham	WHam (LUT)	WHam
	49.2	57.0	971.1

5. RELATED WORK

Machine learning has been applied to improve performance of local descriptor on both large scale descriptor matching [14] and object retrieval [15]. Shakhnarovich [16] proposed BoostPro to find a set of projections for SIFT that are sensitive to similarity pairs used for training. Cai et al. [17] applied linear discriminative projections to SIFT to learn descriptors with low dimensionality. Babenko et al. [18] treated correspondence problem as a classification problem and used boosting to train the classifier. Ozuysal et al. [19] proposed random ferns and to train a Semi-Naive Bayesian Classifier to recognize keypoints directly from their local patches. Locality Sensitive Hashing (LSH) [20] and Spectral Hash (SH) [21, 22] have been proposed to find an efficient binary representation of high-dimensional data maintaining their similarities in the new Hamming space. In contrast to these approaches, our work focuses on binary descriptors and utilizes metric learning to learn a weighted Hamming distance for better matching performance. Meanwhile, a look-up-table strategy is proposed for computing the weighted Hamming distance without sacrificing the computational efficiency of Hamming distance.

6. CONCLUSION

This paper proposes the weighted Hamming distance to improve the matching performance of binary descriptors while preserving its advantage of fast computation. The weighted Hamming distance is learned by minimizing a margin-based ranking loss such that distances of matching pairs are smaller than distances of non-matching pairs by one unit at least. Owing to the proposed look-up-table based strategy, the weighted Hamming distance can be computed as efficient as Hamming distance. Experimental results have shown the effectiveness of the proposed method.

7. ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation of China (61203277,61272394,61005013) and Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation.

8. REFERENCES

[1] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1281–1298, 2012.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," in *International Conference on Computer Vision and Pattern Recognition*, 2012, pp. 510–517.

[3] David Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] Bin Fan, Fuchao Wu, and Zhanyi Hu, "Aggregating gradient distributions into intensity orders: A novel local image descriptor," in *Internation Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2377–2384.

[5] Zhenhua Wang, Bin Fan, and Fuchao Wu, "Local intensity order pattern for feature description," in *International Conference on Computer Vision*, 2011, pp. 603–610.

[6] P.F. Alcantarilla, A. Bartoli, and A. Davison, "KAZE features," in *European Conference on Computer Vision*, 2012, pp. 214–227.

[7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *International Conference on Computer Vision*, 2011, pp. 2564–2571.

[8] Edward Rosten and Tom Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, 2006, pp. 430–443.

[9] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *International Conference on Computer Vision*, 2011, pp. 2548–2555.

[10] Elmar Mair, Gregory D. Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *European Conference on Computer Vision*, 2010, pp. 183–196.

[11] Matthew Brown, Gang Hua, and Simon Winder, "Discriminative learning of local image descriptors," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 43–57, 2011.

[12] Noah Snavely, Steven M. Seitz, and Richard Szeliski, "Photo tourism: Exploring photo collections in 3D," *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 835–846, 2006.

[13] Michael Geosele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz, "Multi-view stereo for community photo collections," in *International Conference on Computer Vision*, 2007.

[14] Christoph Strecha, Alexander M. Bronstein, Michael M. Bronstein, and Pascal Fua, "LDAHash: Improved matching with smaller descriptors," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 66–78, 2012.

[15] James Philbin, Michael Isard, Josef Sivic, and Andrew Zisserman, "Descriptor learning for efficient retrieval," in *European Conference on Computer Vision*, 2010, pp. 677–691.

[16] G. Shakhnarovich, "Learning task-specific similarity," *PhD thesis, MIT*, 2006.

[17] Hongping Cai, Krystian Mikolajczyk, and Jiri Matas, "Learning linear discriminant projections for dimensionality reduction of image descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 338–352, 2011.

[18] Boris Babenko, Piotr Dollr, and Serge Belongie, "Task specific local region matching," in *International Conference on Computer Vision*, 2007.

[19] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 448–461, 2010.

[20] Aristides Gionis, Piotr Indyk, and Rajeev Motwani, "Similarity search in high dimensions via hashing," in *International Conference on Very Large Data Bases*, 1999, pp. 518–529.

[21] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems*, 2009, pp. 1753–1760.

[22] Brian Kulis and Trevor Darrell, "Learning to hash with binary reconstructive embeddings," in *Advances in Neural Information Processing Systems*, 2009, pp. 1042–1050.