Learning Binary Descriptors from Images

Philip Lee, Aditya Srinivas Timmaraju
{philee,adityast}@stanford.edu
Department of Electrical Engineering
Stanford University

***Abstract*—Binary descriptors have become popular for computer vision tasks because of their potential for smart phone applications. However, most binary descriptors have been heuristically hand-crafted. In this paper, we present a methodology to learn sparse binary descriptors from images. A new sampling and comparison pattern is also introduced and its advantages over the existing descriptors are discussed. We show experimental results for the task of matching pairs of images on the Patch dataset, using our descriptor MALCOM (Machine Learned Compact Descriptor). Results indicate that MALCOM's performance surpasses that of FREAK, a state-of-the-art binary descriptor.**

I. Introduction

Local keypoint descriptors are an integral part of many computer vision algorithms such as object matching and recognition. These descriptors are methods to encode information about an image patch around a detected keypoint. Several popular descriptors, such as Scale-Invariant Feature Transform (SIFT) [1], Speeded-up Robust Features (SURF) [2], Binary Robust Invariant Scalable Keypoints (BRISK) [3], and Fast Retina Keypoint (FREAK) [5] have been devised for different purposes. In recent times, binary pattern based descriptors have become popular because of their advantages in computational complexity. These are especially suited for applications that need to run in real-time and/or on a smartphone.

Currently, most binary descriptors rely on hand-crafted, heuristic structures. There is no way to tell whether these are the optimal descriptors for a given task. The major motivation for our project stems from these observations. In this report, we present a framework for using machine learning to devise optimal binary descriptors, given a database of training images. We introduce a new sampling and comparison pattern to arrive at our binary descriptor. By optimizing over an appropriately formulated loss function, we also learn the relative importances corresponding to each dimension in our vector. In this context, Figure 1 briefly outlines the logical flow of our method. We demonstrate results using the learned descriptor MALCOM, compared to the state-of-the-art descriptor FREAK, in relation to an image matching task.

II. Background

In this section, we briefly present previous work on descriptors, with an emphasis on binary descriptors. These differ from general keypoint descriptors in that they describe images patches using only a string of bits. In general, binary keypoint descriptors choose a pattern in the image patch and make pairwise comparisons between mean intensities over predetermined pairs of elements in the pattern. These pairwise comparisons (for instance, whether the first is greater or less than the second) are stored as bits for the given image patch. Given a query image in a recognition/retrieval task, the same pairwise comparisons are used to extract a bit string for each keypoint in the image. As a dissimilarity measure between two bit strings, these methods use the Hamming distance (Bit-wise XOR followed by counting 1s in the resulting string), which is a natural extension of Euclidean distance for the case of binary vectors. One major advantage of using binary descriptors is that Hamming distance computation can be performed very fast, since it can be implemented using XOR and only involves fixed-point arithmetic.

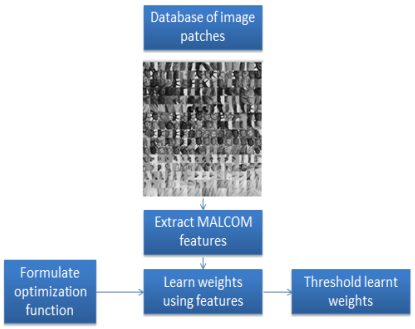

Fig. 1. A logic flow diagram of our approach

A. Previous Work

Descriptors we reference can be broken into three categories. These include SIFT-like descriptors, binary descriptors, and weighted binary descriptors.

1) SIFT-like Descriptors: Scale Invariant Feature Transform (SIFT) was developed by David Lowe and was first published in 1999 [1]. It was one of the first methods for keypoint detection and description. Using Difference of Gaussians as a keypoint detector, it combines sets of orientation histograms into a 128 dimension keypoint descriptor. SIFT has been widely adopted for use in many computer vision tasks, and several offshoots to SIFT have been proposed, such as GLOH, PCA-SIFT, and CARD [9] [10] [11]. However, SIFT remains the most

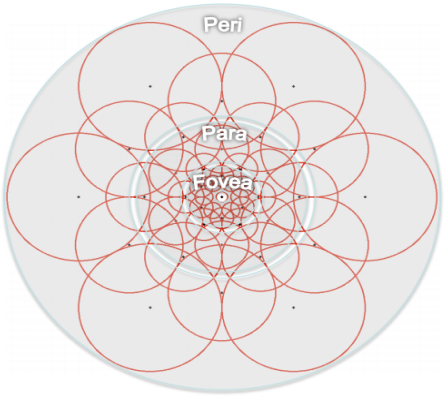

Fig. 2. Illustration of FREAK sampling pattern. Image courtesy [5]

well known keypoint descriptor today. In 2006, H. Bay et al. presented the Speeded-up Robust Features (SURF) descriptor [2]. Designed to be faster than SIFT, it uses Haar-wavelet responses efficiently computed on integral images to approximate similar orientation histograms. These descriptors can be 64 or 128 dimensional vectors.

2) Binary Descriptors: The recent surge in the popularity of smartphones and embedded systems has led to increased demand for even faster and more efficient descriptors. In 2010 M. Calonder, et. al presented the Binary Robust Independent Elementary Features (BRIEF) [4]. These binary feature descriptors are computed by applying Gaussian smoothing kernels over the image patch, followed by performing pairwise comparisons between randomly selected pairs. These descriptors have fast computation time and can be compared efficiently using Hamming distance, as opposed to Euclidean distance for real vectors. This work effectively showed that patches can be described by binary strings and measured the computational advantages for doing so. Other binary descriptors in the same vein have been proposed such as Binary Robust Invariant Scalable Keypoints (BRISK) [3], Oriented Fast and Rotated BRIEF (ORB) [7], and Fast Retina Keypoint (FREAK) [5]. The main difference between these approaches are the small modifications in filters applied and pairwise comparisons performed. ORB is designed to be invariant to rotation and robust to noise by taking into account the orientation of the keypoint. BRISK is made to be invariant to scale and rotation by using circular sampling patterns instead of randomly chosen ones. FREAK emulates the sampling pattern of the human retinal system in a pattern shown in Figure 2.

3) Learned Binary Descriptors: Although binary descriptors suffice in many practical applications, they are still not as robust or effective as SIFT-like descriptors. There have been a few recent studies in applying machine learning techniques to optimize the efficacy of binary descriptors. Bin boost is an approach that uses boosting to combine advantages of different binary descriptors into a single feature vector [12]. Another approach is using a Fisher Discriminant method to project the image into a dimension with highest discrimination [13]. Of particular note to our project is work done by B. Fan et. al, learning weights for binary descriptors [6]. In this paper, they begin with established descriptors and learn a margin classifier to discriminate between pairs of patches that are matches against pairs that are not. However, this classifier supplies weights to each bit position, losing some of the main computational advantages of binary descriptors.

B. Our Contributions

We take inspiration from the idea of finding weighted binary descriptors, but deviate in two major fashions. First, instead of keeping the weights (and adding computational complexity to the binary descriptors) we only use weights as a measure of importance in a bit position and threshold positions by learned weights. So, we do not need to resort to using weighted Hamming distance. Second, instead of using an already established descriptor and assigning weights to its bit positions, we propose a new sampling pattern and comparison structure. As will be seen in the following section, this structure captures a novel way to encode more comparisons than those needed to be stored in our descriptor. Once we learn the weight vector, we retain only the bit positions that are most correlated with our model (those with the weights that are found to be the highest). Our approach is discussed in more detail in the following section.

III. Technical Approach

This section first describes a high level overview of our approach. It then gives more details about the specific methodologies we adopted.

A. Summary of Technical Solution

A major difference between various binary descriptors lies in the sampling pattern used and comparisons performed. In our approach, instead of using a relatively few, heuristically chosen comparisons from the sampling pattern as in current binary descriptors, we choose a pattern with a large number of comparisons with the intention of removing ones found to be less important. Thus, our approach is to enumerate many comparisons and use a machine learning approach to prune them down. To accomplish this, we used a multi-level square comparison pattern as shown in Figure 3. We label our descriptor, MALCOM (Machine Learned Compact descriptor). We then compute MALCOM descriptors for image patches from the Patch dataset [8]. We used these descriptors to learn a margin classifier to distinguish between matching pairs of patches and non-matching pairs. Finally, we thresholded the weights learnt from our classifier to enforce sparsity (as it is, most weights are found to be significantly low) and eliminate the extra computational cost of a floating point multiplication. Each of these steps is described in further detail in the following subsections.

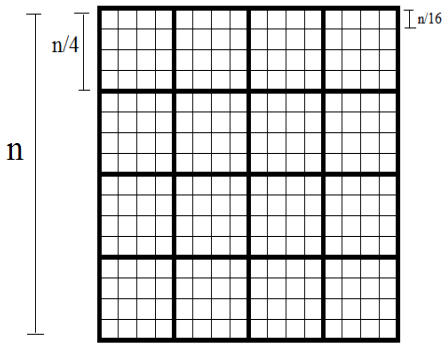

Fig. 3. Illustration of the Multi-Level Sampling Pattern. The image is divided into large and small blocks. Pairwise comparisons are made between mean intensities over these blocks.

B. Sampling Pattern

We developed a multi-level sampling pattern to efficiently create as many meaningful comparisons as possible. First, an image patch is divided into 4 x 4 blocks. We denote these as large blocks. The average intensity over each of these blocks is computed. At the second level, each of these large blocks is itself further divided into 4 x 4 sub-blocks, labeled small blocks. The average intensity over each of these small blocks is computed. Thus, there are 16 large blocks, each divided into 16 small blocks. This constitutes a total of 256 small blocks (16 in each large block) over the full image patch. The sampling pattern is depicted in Figure 3. We then make pairwise comparisons in the following three ways. First, we perform a comparison for each pair of large blocks. This constitutes 120 comparisons and results in 120 bits. Second, for each large block, we compare the large block mean intensity to that of each of its constituent small block means. This results in a total of 256 bits. Finally, within a large block, we make pairwise comparisons between all small blocks. This is results in 1920 bits. Thus, our final binary string is 2296 bits long. These calculations are summarized in Table I. It is important to note that all possible pairs of small and large blocks would constitute $\binom{(16+256)}{2}$ = 36856 bits. This is more than an order of magnitude larger. Because of our chosen framework, our method implicitly encodes some of this transitive information, i.e., comparisons between small blocks in different large blocks as well. To see this, we highlight that we have comparisons between a large block and its constituent small blocks, and also between large blocks, which can lead us to deduce information about the relationship between small blocks belonging to different large blocks. Thus, much of the 36856 brute force vector information would anyway be redundant. Our approach encompasses a novel way to shrink it by an order of magnitude.

C. Margin Classifier

Using our sets of 2296 bit keypoint descriptors found for our training set, we learned a weight vector w for a margin classifier similar to that in [6]. The main motivation for assigning different weights to different binary positions is that non-matching image pairs can be made to have higher distance scores than matching image pairs. Owing to the fact that matching is usually done by searching for the nearest neighbor, the absolute value of the distance scores is less relevant than the relative ranking among them. These weights can be used to form a marginal classifier. Additionally, the weights in this learned weight vector w reflect the relative contributions of each of the bit positions. The weights are learned in the following manner:

TABLE I
DIMENSIONS OF EACH CATEGORY

Type of comparison	Calculation	# Comparisons
Large-Large	$\binom{16}{2}$	120
Large-Small	$16 * 16$	256
Small-Small	$16 * \binom{16}{2}$	1920
Total	$120 + 256 + 1920$	2296

We first denote as $WHam(X_1, X_2)$, the weighted hamming distance between bit vectors X_1 and X_2 as follows,

$$WHam(X_1, X_2) = \sum_{i=1}^{n} w_i * (X_{1,i} \oplus X_{1,j}) \tag{1}$$

A margin-one learning objective can be formulated using the following set of constraints:

$$\begin{aligned} WHam(X_1, X_2) < WHam(Y_1, Y_2) - 1, \\ \forall (X_1, X_2) \in M, (Y_1, Y_2) \in N \end{aligned} \tag{2}$$

where M and N are the sets containing matching pairs and non-matching pairs of images respectively. From this, the empirical loss function can be formulated [6] as,

$$\begin{aligned} l(w) = \sum_{(X_1,X_2)\in M} \sum_{(Y_1,Y_2)\in N} \\ max\{WHam(X_1, X_2) - WHam(Y_1, Y_2) + 1, 0\} \end{aligned} \tag{3}$$

The optimization problem is formulated as,

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

where, the regularization term is set to $R(w) = ||w||_1$ and $\lambda = 100$ in our experiment. We chose an L-1 norm for regularization to encourage sparsity in the vector. For our project, we formulated this as a convex optimization problem and used the convex program software package CVX to solve it [14] [15].

D. Thresholding Weights

As discussed in the previous sections, using weights for bit positions negates the computational benefit to binary descriptors. So, instead of retaining the added computational complexity of floating point multiplications of learned weights, we instead use weights as only a measure of relevance for each bit position and threshold the dimensions based on weights. This is equivalent to setting a weight

Fig. 4. Example of a Liberty patch

to be 1 if it is above our threshold and 0 if below. This has two main advantages. It recovers the ability of binary descriptors to use binary arithmetic and allows us to enforce sparsity on our feature vectors. In the following section, we show results for different levels of sparsity enforced (with different thresholds) and also those obtained retaining the weights.

IV. Experimental Results

We have tested our new binary descriptor, MALCOM on the Patch Dataset [8]. The Patch Dataset was specifically designed to help in the evaluation of local image descriptors and hence is suited for this purpose. The dataset contains over 400k patches each from categories named Liberty, Notre Dame and Yosemite. A sample patch from the Liberty set is depicted in Fig 4. These patches were obtained using Harris Corner detectors and Difference of Gaussian detectors that have been corrected for scale. Each patch is a 64 x 64 resolution image. In our experiments, we learned the weights using images belonging to one category and tested it on images from the other two categories. We repeated this for all combinations, which sums to a total of six different settings. In each setting, we used 6k patch pairs for training and 45k patch pairs for testing.

We tabulate the results from two different methods. In the first one (MALCOM), we use the full weight vector and a weighted Hamming distance as the dissimilarity measure. In the second method (H-MALCOM 128), we retain only the top 128 bits (as indicated from the learned weights) and use only Hamming distance, instead of a weighted Hamming distance. We choose to retain 128 bits because it is an integral multiple of the word length in most modern processors.

Table II summarizes the results from our methods and compares it to the performance of FREAK, the state-of-the-art binary descriptor, in each of the six settings. Figures 5-10 depict the ROC (Receiver Operating Characteristic) curves for each setting. In these plots, the curve for H-MALCOM (16 bit) corresponds to the results obtained using only the 16 most important bits in the descriptor and H-MALCOM (256-bit) uses the top 256 bits.

We see that our descriptor outperforms FREAK in all configurations. We attribute this to a few key characteristics of our chosen descriptor. The first is the nature of multi-scale comparisons, which inherently encodes comparisons between small blocks in different larger blocks. Second, the learning function identifies the most discriminative dimensions. Importantly, as is done in previous works, we do not rely on heuristics or other simpler arguments for maximizing this discriminative information.

TABLE II
Area under the ROC Curves for different settings

Train/Test Configuration	FREAK	MALCOM	H-MALCOM 128
Liberty/Notre Dame	0.838	0.934	0.932
Liberty/Yosemite	0.829	0.924	0.922
Notre Dame/Liberty	0.815	0.913	0.907
Notre Dame/Yosemite	0.829	0.926	0.921
Yosemite/Liberty	0.809	0.896	0.892
Yosemite/Notre Dame	0.863	0.935	0.933

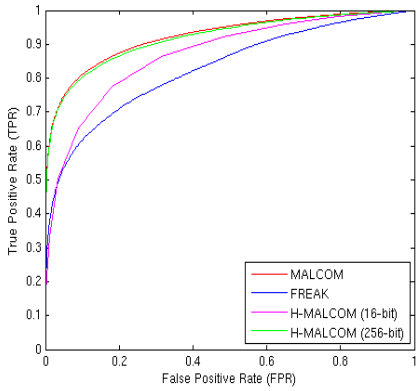

Fig. 5. ROC Curve for Liberty/Yosemite (Train/Test) Configuration

V. Conclusion

There are several positive outcomes from this project. We have designed a new sampling pattern for performing pairwise comparisons to arrive at the binary descriptor. The multi-scale structure inherently captures transitive comparisons between small blocks in different big blocks. The learned weight vector is sparse. Even with as few as 128 bits, our descriptor H-MALCOM 128 outperforms FREAK, the current state-of-the art binary descriptor. Also, compared to the method in [6], this approach does not need to use a weighted Hamming distance and hence does not accrue additional computational complexity introduced by including the weights. Significantly, we underscore here that the weights by themselves are not as vital as the important dimensions learned in our descriptor. This is evident from the relatively insignificant drop in performance from MALCOM to H-MALCOM 128 in Table I.

Acknowledgment

The authors would like to thank project mentor Dr. Alexandre Alahi for guidance in introducing us to this topic and support throughout the project. In addition, we thank Prof. Silvio Savarese and the CS 231A Teaching staff for giving us the chance to learn about computer vision and work on this project.

References

[1] David G. Lowe ,"Distinctive image features from scale-invariant keypoints" International Journal of Computer Vision 60.2 (2004): 91-110.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded-up robust features" in Computer Vision - ECCV 2006. Springer Berlin Heidelberg, 2006. 404-417.

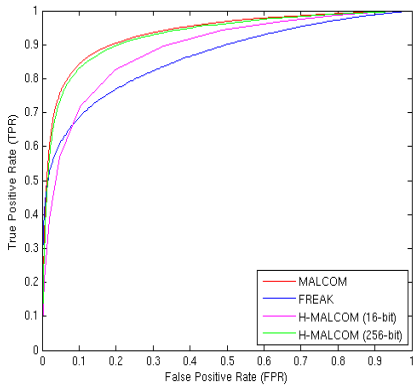

Fig. 6. ROC Curve for Liberty/Notre Dame (Train/Test) Configuration

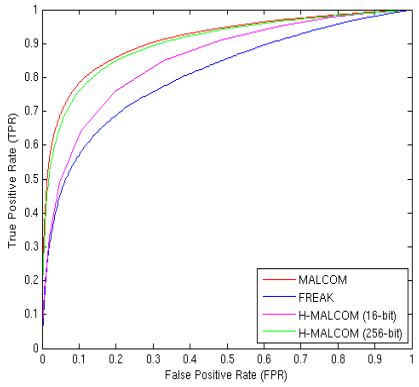

Fig. 7. ROC Curve for Notre Dame/Liberty (Train/Test) Configuration

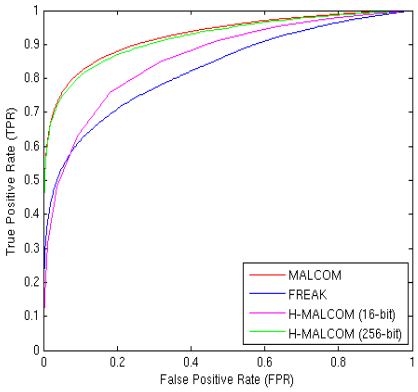

Fig. 8. ROC Curve for Notre Dame/Yosemite (Train/Test) Configuration

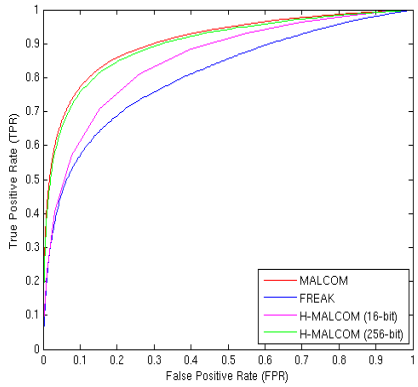

Fig. 9. ROC Curve for Yosemite/Liberty (Train/Test) Configuration

[3] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints" IEEE International Conference on Computer Vision (ICCV), 2011.

[4] Michael Calonder, et al. "BRIEF: Binary robust independent elementary features" in Computer Vision - ECCV 2010. Springer Berlin Heidelberg, 2010. 778-792.

[5] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst, "FREAK: Fast retina keypoint" IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[6] Bin Fan, et al. "Learning weighted Hamming distance for binary descriptors" IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013.

[7] Ethan Rublee, et al, "ORB: An efficient alternative to SIFT or SURF" IEEE International Conference on Computer Vision (ICCV) 2011.

[8] Matthew Brown, Gang Hua, and Simon Winder, "Discriminative learning of local image descriptors" Pattern Analysis and Machine Intelligence, IEEE Transactions on 33.1 (2011): 43-57.

[9] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors" Pattern Analysis and Machine Intelligence, IEEE Transactions on, (2005): 1615-1630.

[10] Yan Ke, and Rahul Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors" Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[11] M. Ambai and Y.Yoshida, "CARD: Compact and real-time descriptors" In Computer Vision, 2011 IEEE Computer Society Conference on. IEEE, 2011.
M. Ambai, and Y. Yoshida, "CARD: Compact and real-time descriptors" IEEE International Conference on Computer Vision (ICCV), 2011.

[12] Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit, "Boosting binary keypoint descriptors" IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2874-2881.

[13] Tomasz Trzcinski, and Vincent Lepetit, "Efficient discriminative projections for compact binary descriptors" in Computer Vision - ECCV 2012. Springer Berlin Heidelberg, 2012. 228-242.

[14] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx, September 2013.

[15] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, Lecture Notes in Control and Information Sciences, Springer, 2008. http://stanford.edu/~boyd/graph_dcp.html.

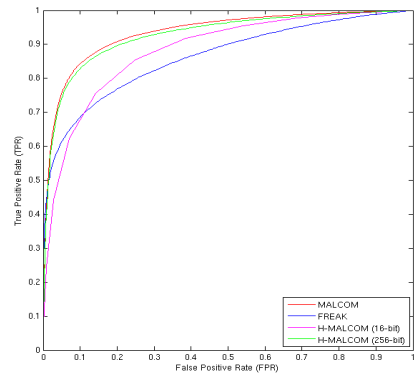

Fig. 10. ROC Curve for Yosemite/Notre Dame (Train/Test) Configuration