LATCH: Learned Arrangements of Three Patch Codes

Gil Levi
The Open University of Israel
gil.levi100@gmail.com

Tal Hassner
The Open University of Israel
USC / Information Sciences Institute
hassner@openu.ac.il

Abstract

We present a novel means of describing local image appearances using binary strings. Binary descriptors have drawn increasing interest in recent years due to their speed and low memory footprint. A known shortcoming of these representations is their inferior performance compared to larger, histogram based descriptors such as the SIFT. Our goal is to close this performance gap while maintaining the benefits attributed to binary representations. To this end we propose the Learned Arrangements of Three Patch Codes descriptors, or LATCH. Our key observation is that existing binary descriptors are at an increased risk from noise and local appearance variations. This, as they compare the values of pixel pairs: changes to either of the pixels can easily lead to changes in descriptor values and compromise their performance. In order to provide more robustness, we instead propose a novel means of comparing pixel patches. This ostensibly small change, requires a substantial redesign of the descriptors themselves and how they are produced. Our resulting LATCH representation is rigorously compared to state-of-the-art binary descriptors and shown to provide far better performance for similar computation and space requirements.

1. Introduction

The ability to effectively represent local visual information is key to a very wide range of computer vision applications. These applications range from image alignment, which requires that local image descriptors be accurately matched between different views of the same scene, to image classification and retrieval, where massive descriptor collections are repeatedly scanned in order to locate the ones most relevant to those of a query image. Consequently, computer vision research has devoted substantial efforts to develop and fine-tune these representations.

At the core of the problem is the challenge of extracting local representations at keypoints, typically distributed sparsely over an image, in a manner which is both discriminative and invariant to various image transformations. Additional requirements, often as important if not more, are

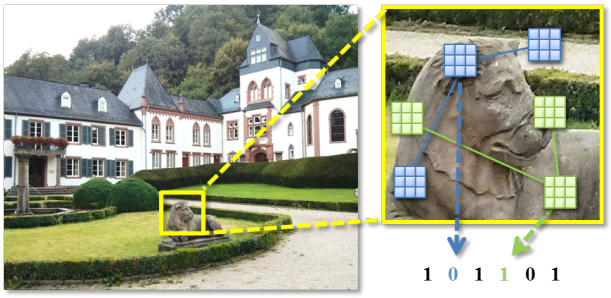


Figure 1. **Visualization of the LATCH descriptor.** Given an image patch centered around a keypoint, LATCH compares the intensity of three pixel patches in order to produce a single bit in the final binary string representing the patch. Example triplets are drawn over the patch in green and blue.

that a representation be efficient in terms of the computational costs required to produce it, the space required to store it and the time required to search for matching descriptors in large descriptor repositories.

Over the past two decades, several distinct approaches for designing such descriptors have emerged. Two noteworthy designs are the distribution-based representations and the binary descriptors. Distribution-based descriptors, which include the successful SIFT [17] and HOG [8] representations, represent visual information using distributions of image measurements (e.g., gradients, gradient orientations, etc.). Though proven highly effective in an ever widening range of applications, their main drawbacks are their size, the time required to produce them, and the challenges associated with efficiently searching through large numbers of such descriptors [14].

Binary descriptors, on the other hand, were designed with an emphasis on minimizing computational and storage costs [2, 6, 15, 16, 24, 26, 28, 29]. These methods represent image patches using a (typically short) binary string, commonly computed by sampling and comparing pixels in the patch; different methods advocating different sampling strategies or other methods for increasing the descriptors discriminate power (e.g. boosting, discriminant projections). Though binary representations may not be as descriptive as their histogram counterparts, they make up for

this shortcoming in their compact size, efficient computation, and the ability to quickly compare descriptor pairs using few processor-level instructions.

The representation presented here belongs to the latter family of descriptors. Our work is motivated by the long-standing observation that the act of sampling pixel pairs in order to compute each binary value in the representation is sensitive to noise and other changes in local appearances. Previous representations have addressed this problem by offering a number of alternative smoothing operations which should be performed before the pixel values are sampled. Though this alleviated some of the problem, the unfortunate side effect of smoothing is, of course, the loss of information. This is particularly crucial in high-frequency regions of the image – precisely where key points are detected, and where these representations are applied.

We offer an alternative approach based on the simple notion of comparing pixel patches rather than individual pixel values (illustrated in Fig. 1). By comparing patches, visual information with more spatial support is considered for each of the descriptor's bits, and their values are therefore less sensitive to noise. We describe a patch-triplet based approach, in which triplets of patches are compared in order to set the binary values of the representation. Informative triplet arrangements are learned beforehand using labeled training data. Thus, triplet arrangements are ordered by their contribution to the successful classification of patches as being either similar or not while refraining from selecting highly correlated triplets. The most effective arrangements of patch triplets are then used to sample and compare patches whenever the descriptor is computed.

The resulting representation, appropriately dubbed LATCH (Learned Arrangements of Three patCH codes), is evaluated extensively and shown to outperform existing alternatives by a wide margin, at the cost of a minor increase in the run-time computational requirements of extracting the descriptor. To summarize, this paper makes the following contributions.

- We propose a novel binary descriptor design, intended to provide improved stability and robustness than existing related descriptors.
- We show how effective descriptors can be generated by off-line, supervised learning of discriminative patch arrangements.
- Extensive quantitative results and qualitative applications compare the capabilities of our LATCH representation with existing descriptors. These show LATCH to outperform other representations of its kind, significantly narrowing the performance gap between binary descriptors and histogram based methods.

Our implementation has been incorporated into the OpenCV library and is publicly available. Please see project page[1] for more details.

2. Related Work

The development of local image descriptors has been the subject of immense research, and a comprehensive review of related methods is beyond the scope of this work. For a recent survey and evaluation of alternative binary interest point descriptors, we refer the reader to [14]. Here, we only briefly review these and other related representations.

Binary descriptors. Binary keypoint descriptors were recently introduced in answer to the rapidly expanding sizes of image data sets and the pressing need for compact representations which can be efficiently matched. One of the first of this family of descriptors was the Binary Robust Independent Elementary Features (BRIEF) [6]. BRIEF is based on intensity comparisons of random pixel pairs in a patch centered around a detected image key point. These comparisons result in binary strings that can be matched very quickly using a simple XOR operation. As BRIEF is based on intensity comparisons, instead of image gradient computations and histogram pooling of values, it is much faster to extract than SIFT-like descriptors [17]. Furthermore, by using no more than 512 bits, a single BRIEF descriptor requires far less memory than its floating point alternatives.

Building upon BRIEF's design and matching method, the Oriented fast and Rotated BRIEF (ORB) descriptor [24] adds rotation invariance by estimating a patch orientation based on local first order moments within the patch. Another innovation proposed by [24] is the use of a unsupervised learning in order to select pixel pairs, rather than the random sampling of BRIEF.

Rather than random sampling or unsupervised learning of pairs, the Binary Robust Invariant Scalable Keypoints (BRISK) [16] use hand-crafted, concentric ring-based sampling patterns. BRISK uses pixel pairs with large distances between them to compute the patch orientation, and pixel pairs separated by short distances to compute the values of the descriptor itself, again, by performing binary intensity comparisons on pixel pairs. More recently, inspired by the retinal patterns of the human eye, the Fast REtinA Keypoint descriptor (FREAK) was proposed. Similarly to BRISK, FREAK also uses a concentric rings arrangement, but unlike it, FREAK samples exponentially more points in the inner rings. Of all the possible pairs which may be sampled under these guidelines, FREAK, following ORB, uses unsupervised learning to choose an optimal set of point pairs.

Similar to BRIEF, the Local Difference Binary (LDB) descriptor was proposed in [33, 34] where instead of comparing smoothed intensities, mean intensities in grids of

[1] www.openu.ac.il/home/hassner/projects/LATCH

2×2, 3×3 or 4×4 were compared. In addition to the mean intensity values, LDB compares the mean values of horizontal and vertical derivatives, amounting to 3 bits per comparison. Building upon LDB, the Accelerated-KAZE (A-KAZE) descriptor was suggested in [3] where in addition to presenting a feature detector, the authors also suggest the Modified Local Difference Binary (M-LDB) descriptor. M-LDB uses the A-KAZE detector estimation of orientation for rotating the LDB grid to achieve rotation invariance and uses the A-KAZE detector's estimation of feature scale to sub-sample the grid in steps that are a function of the feature scale.

A different design approach was proposed by [26]. Their LDA-Hash extracts SIFTs from an image, projects them to a discriminant space and then thresholds the projected descriptors to obtain binary representations. Producing LDA-Hash requires extracting SIFT descriptors, making it slower than its pure binary alternatives. To alleviate some of this computation, DBRIEF [29] projects patch intensities directly. Projections are computed as linear combinations of few, simple filters from a given dictionary. The BinBoost representation of [15, 28] learns a set of hash functions corresponding to each bit in the final descriptor. Hash functions are learned using boosting and implemented as sign operations on a linear combination of non linear week classifiers. Finally, PR-proj [25] use a combination of learning-based methods and dimensionality reduction techniques to reduce image gradient measurements into compact and effective binary representations.

These last representations, LDA-Hash, DBRIEF, BinBoost and PR-proj, all obtain binary representations following application of filter combinations or floating-point descriptor extraction. Thus, though they show improved performance over the original binary descriptors, they are all far more expensive computationally and so may be unsuitable in many practical applications.

Unlike these methods, our own uses efficient patch comparisons directly. Unlike the earlier representations (i.e. BRIEF,ORB,BRISK and FREAK), rather than comparing pairs of pixels, we compare *triplets of pixel patches* thereby providing more spatial support for each comparison. This provides more information at each comparison, making the binary values more robust to various sources of noise. Doing so also requires redesigning the descriptor itself. Finally, in contrast to the unsupervised learning of arrangements proposed by ORB, we use supervised learning to obtain efficient patch combinations.

Local binary patterns. In a separate line of work, the Local Binary Patterns (LBP) were proposed as global (whole image) representation by [22, 23]. Since then, they have been successfully applied to many image classification problems, most notably of texture and face images (e.g., [1] and [21]).

LBP produces for each pixel in the image a (typically very short) binary string representation. In fact, to our knowledge, 8-bit strings or less were employed in all applications of LBP. These bits, similarly to the binary descriptors, are set following binary comparisons between image pixel intensities. In the original LBP implementation, these bits were computed by using a pixel's value as a threshold, applied to its eight immediate spatial neighbors, and taking the resulting zero/one values as the 8-bit string. By using only 8-bits, each pixel is thus represented by a code in the range of $[0..255]$ (or less, in some LBP variations), which are then pooled spatially in a histogram in order to represent image portions or entire images.

Our work is related to a particular LBP variant, the Three-Patch LBP (TPLBP) [31, 32], which was shown to be an exceptionally potent global representation for face images [10]. Unlike previous LBP code schemes, TPLBP computes 8-bit value codes by comparing not the intensities of pixel pairs, but rather the similarity of three pixel patches. Specifically, for every pixel in the image, TPLBP compares the pixel patch centered on the pixel, with eight pixel patches, evenly distributed on a ring at radius r around the pixel. A single binary value is set following a comparison of the center patch to two patches, spaced α degrees away from each other along the circle. A value of 1 represents the central patch being closer (in the SSD sense) to the first of these two patches, 0 otherwise.

The TPLBP codes, though similar in spirit to the LATCH descriptor presented here, are different from it in several important aspects. Technically, TPLBP use a hand tailored, parameter controlled, limited sampling scheme, where a single anchor patch (the central patch) is compared again and again with a limited number of patch pairs at specific relative positions controlled by the ring radius r and the angle between patches α. Our proposed LATCH, on the other hand, can potentially consider *any* arrangement of three patches for this purpose. Moreover, LATCH *learns* which arrangements are optimal from training data, rather than being hand-crafted.

More important, however, is the conceptual difference: LATCH is designed as a (sparse) keypoint descriptor, rather than a per-pixel code intended for pooling over image regions. To our knowledge, no previous work has considered using the design insights of TPLBP to represent key points.

3. Method

We begin with a review of binary descriptor design. Let W be a detection window, an image portion of fixed, predetermined size, centered on a detected image key point. A binary descriptor $\mathbf{b}_W$ is formed by considering an ordered set $S = \{\mathbf{s}_t\}_{t=1...T} = \{[\mathbf{p}_{t,1}, \mathbf{p}_{t,2}]\}_{t=1...T}$ of T pairs of sampling coordinates, $\mathbf{p}_{t,1} = (x_{t,1}, y_{t,1})$ and $\mathbf{p}_{t,2} = (x_{t,1}, y_{t,2})$, given in W's coordinate frame. The selection of

Descriptor	Running time (ms)
SIFT [17]	3.29
SURF [4]	2.11
LDA-HASH [26]	5.03
LDA-DIF [26]	4.74
DBRIEF [29]	8.75
BinBoost [15, 28]	3.29
BRIEF [6]	0.234
ORB [24]	0.486
BRISK [16]	0.059
FREAK [2]	0.072
A-KAZE [3]	0.069
LATCH	0.616

Table 1. **Run time analysis**. Time measured in milliseconds for extracting a single local patch descriptor. Notice that LATCH only slightly slower than some of the popular binary descriptors and is an order of magnitude faster than the slower histogram and learning-based representations.

values for S is performed beforehand, either randomly (e.g., BRIEF [6]), manually (BRISK [16]), or is automatically learned from training data (ORB [24] and FREAK [2]).

Each index t is typically associated not only with a pair of coordinates in W, but also with a pair of Gaussian smoothing kernels, $\sigma_t = (\sigma_{t,1}, \sigma_{t,2})_{t=1...T}$. These are applied separately to W, in the pixel coordinates given by $\mathbf{s}_t$, before being sampled. Thus, for each sampling pair $\mathbf{s}_t$, the smoothed intensities at the two sampling points $\mathbf{p}_{t,1}$ and $\mathbf{p}_{t,2}$, are compared and a single bit is set according to:

$$f(W, \mathbf{s}_t, \sigma_t) = \begin{cases} 1 & \text{if } W(\mathbf{p}_{t,1}, \sigma_{t,1}) > W(\mathbf{p}_{t,2}, \sigma_{t,2}) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $W(\mathbf{p}_{t,1}, \sigma_{t,1})$ (similarly $W(\mathbf{p}_{t,2}, \sigma_{t,2})$) is the value of the image window W at coordinates $\mathbf{p}_{t,1}$ ($\mathbf{p}_{t,2}$) smoothed by a Gaussian filter with standard deviation $\sigma_{t,1}$ ($\sigma_{t,2}$). The final binary string $\mathbf{b}_W$, produced for image window W, is defined by

$$\mathbf{b}_W = \sum_{1 \le t \le T} 2^t f(W, \mathbf{s}_t, \sigma_\mathbf{t}) \tag{2}$$

3.1. From pixel pairs to patch triplets

As previously mentioned, the pixel pairs sampling strategy presented above, though efficient, can be susceptible to noise as each bit relies on the values of two specific pixels. Though pre-smoothing can alleviate some of this problem, it can also result in the loss of information particularly at high frequency regions where key points are often detected. As a means of ameliorating this, we propose comparing pixel patches rather than pixels. Doing so, however, requires changing how each bit's value is set and in particular, defining a binary relation between pixel patches. This is achieved by using three-way patch comparisons.

Specifically, we consider $t = 1 \dots T$ pixel patch *triplets*, adding the location of an "anchor" patch and redefining S as $\hat{S} = \{\hat{\mathbf{s}}_t\}_{t=1...T} = \{[\mathbf{p}_{t,a}, \mathbf{p}_{t,1}, \mathbf{p}_{t,2}]\}_{t=1...T}$. Each of the pixel coordinates, $\mathbf{p}_{t,a}$, $\mathbf{p}_{t,1}$, and $\mathbf{p}_{t,2}$ provides the location of the central pixel in patches of size $k \times k$ pixels, denoted by $\mathbf{P}_{t,a}$, for the anchor patch, and $\mathbf{P}_{t,1}$, and $\mathbf{P}_{t,2}$ for its "companion" patches. We then evaluate the similarity of the anchor patch $\mathbf{P}_{t,a}$ to its two companions, by computing their Frobenious norm. Thus, the single binary value is produced by revising function f as follows:

$$g(W, \hat{\mathbf{s}}_t) = \begin{cases} 1 & \text{if } |||\mathbf{P}_{t,a} - \mathbf{P}_{t,1}||_F^2 > ||\mathbf{P}_{t,a} - \mathbf{P}_{t,2}||_F^2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Finally, the binary string $\mathbf{b}_W$ is defined by replacing function f with g in Eq. 2. Importantly, unlike previous binary descriptors, this scheme samples image intensities without pre-smoothing. Though we have experimented by adding smoothing this seemed to provide little benefit. Not surprisingly, the related TPLBP codes of [32] also did not perform such smoothing, presumably for the same reasons.

3.2. Learning patch triplet arrangements

Even small detection windows W give rise to a huge number of possible triplet arrangements. Considering that only a small number T of bits is typically required (in practice, no more than 256), we must therefore consider which of the many possible triplet arrangements should be employed. Here, rather than taking one of the three approaches described by existing binary descriptors (Sec. 2), we propose our own selection criteria.

Specifically, we use the data-set introduced in [5]. It consists of three separate collections: Liberty, Notre Dame, and Yosemite. Each of these contains over 400k local image windows that were extracted around multi-scale Harris corner detections [12]. Pairs of these windows, extracted from different images in each collection, were labeled as being "same" (the two windows present the same physical scene point, viewed from different viewpoints or viewing conditions) or "not-same". These labels were obtained by employing multi-view stereo to form correspondences between different images in each collection. These windows were then partitioned into 500k pairs of which half are labeled as same and half not-same.

We form 56k patch triplet arrangements, by random selection of the pixel coordinates $\mathbf{p}_{t,a}$ of the anchor patch, and the coordinates $\mathbf{p}_{t,1}$ and $\mathbf{p}_{t,2}$ of its two companion patches ($t = 1 \dots T = 56,000$). We then evaluate each of these T arrangements over all the window pairs in the benchmark, giving us 500k bits per arrangement. We define the quality of an arrangement by summing the number of times it correctly yielded the same binary value for "same" labeled

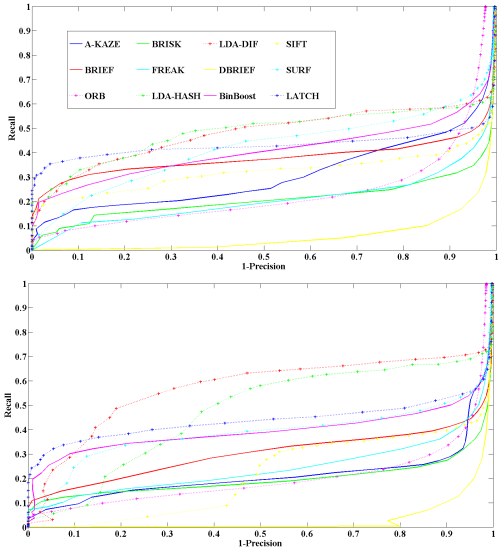


Figure 2. **Oxford benchmark Recall vs. 1-precision curves.** Top: Bikes results (blur) ; Bottom: Leuven results (lightning). Evidently, LATCH outperforms all methods, except those that are an order of a magnitude slower. Notice LATCH's superior performance at the high precision section of the graph.

pairs and different values for "not-same" labeled pairs.

Arrangement selection based on this criteria may result in highly correlated arrangements being selected. To prevent this, following [2, 24], we add arrangements incrementally, skipping over those with responses highly correlated to previously selected arrangements. Specifically, a candidate arrangement is selected if its absolute correlation with all previously selected arrangements is smaller than a threshold τ. In our experiments, this value was set to $\tau = 0.2$ and left unchanged.

We note that others have also used the data-set from [5] for the purpose of learning binary descriptors (e.g. [15, 26, 28, 29]). However, those methods differ from the one proposed here as they do not learn optimal arrangements for pixel comparison, but instead learn optimal projections or linear/non-linear filters to apply to these patches. The method presented here is simpler, yet provides comparative, or even better performance, as we later show.

4. Experimental results

Our LATCH extraction routine is implemented in C++ using OpenCV 2.0 for image processing operations. Unless otherwise noted, we used 32-byte LATCH descriptors with 7×7 patches. Detection windows are 48×48 pixels centered on keypoints. Our tests use the efficient C++ descriptor implementations available from OpenCV or from their various authors, with parameter values left unchanged.

4.1. Empirical results

Comparisons are provided using a wide range of relevant alternative methods. These include the "pure"-binary descriptors: BRIEF [6], ORB [24], BRISK [16], FREAK [2] and A-KAZE [3]. We additionally provide results comparing LATCH to the more computationally expensive LDA-Hash [26], DBRIEF [29] and BinBoost [15, 28] representations. Finally, the performances of SIFT [17] and SURF [4] are also provided.

We note that the original BRIEF descriptor is not invariant to rotations where all others are. To compensate for this, in all our tests we used a slightly modified version of BRIFT: The descriptor was extracted at SIFT keypoints, with the image rotated around each keypoint according to the orientation assigned to it by the detector. This improved its performance compared to some of the more recent descriptors. To avoid confusion, below we refer to this representation as steered BRIEF, or SBRIEF.

We used two standard benchmarks for our tests: the Oxford [18, 19] and the Learning Local Image Descriptors [5] benchmarks. Our tests employ the test protocols associated with these benchmarks. We additionally provide a range of tests designed to evaluate the contribution and effect of various design aspects of our LATCH descriptor.

Run times. We begin by comparing the computational costs associated with extracting the various descriptors used in our experiments. The time (ms) required to extract a single descriptor were averaged over 250K patches of different scale and orientation, taken from various images. Measurements were performed on an Intel Core i7 laptop with 16.0 GB of memory, running 64-bit Microsoft Windows 8.1.

Table 1 summarizes the measured running times. The substantial difference between the time required to extract the pure binary descriptors, including our own LATCH, and descriptors based on floating point values is clearly evident. In particular, LATCH requires an order of magnitude less time than some of these alternatives.

Oxford data-set. Originally described by [18, 19] this set has since become the standard for evaluating descriptor design capabilities, and in particular, the capabilities of the binary descriptors discussed here (see, e.g., [2, 3, 6, 16]).

The Oxford data-set comprises of eight image sets, each with six images presenting increasing appearance variations. The appearance variations modeled by the benchmark sets are: zoom and rotation (the Boat and Bark sets), planar perspective transformations (view-point changes in the Graffiti and Wall sets), lightning changes (the Leuven set), JPEG compression (the UBC set), and increasing degrees of blur (the sets Bikes and Trees).

For each set, we compare the first image against each of the remaining five and check for correspondences. Performance is measured using the code from [18, 19][2], which computes recall and 1-precision using known ground truth

[2] www.robots.ox.ac.uk/~vgg/research/affine

Descriptor	Bark	Bikes	Boat	Graffiti	Leuven	Trees	UBC	Wall	Average
SIFT [17]	0.077	0.322	0.080	0.127	0.130	0.047	0.130	0.138	0.131
SURF [4]	0.071	0.413	0.088	0.133	0.300	0.046	0.268	0.121	0.180
LDA-HASH [26]	0.199	0.466	0.269	0.155	0.303	0.110	0.393	0.268	0.270
LDA-DIF [26]	0.197	0.472	0.278	0.170	0.435	0.101	0.396	0.260	0.289
DBRIEF [29]	0.000	0.025	0.001	0.008	0.010	0.001	0.031	0.002	0.010
BinBoost [15, 28]	0.055	0.344	0.083	0.132	0.338	0.037	0.217	0.119	0.166
BRIEF [6]	0.055	0.353	0.050	0.102	0.227	0.060	0.178	0.141	0.146
ORB [24]	0.032	0.208	0.048	0.062	0.118	0.027	0.121	0.050	0.083
BRISK [16]	0.015	0.138	0.026	0.071	0.161	0.018	0.131	0.038	0.075
FREAK [2]	0.019	0.145	0.034	0.101	0.194	0.026	0.147	0.041	0.089
A-KAZE [3]	0.022	0.326	0.005	0.048	0.138	0.027	0.144	0.048	0.095
LATCH	0.065	0.415	0.057	0.119	0.374	0.082	0.215	0.175	0.188

Table 2. **Oxford benchmark results.** Numerical results summarizing area under the recall vs. 1-precision curve for the eight subsets of the Oxford set. Results for the much larger, floating point, histogram based descriptors are presented separately. Higher results are better. In total, LATCH outperforms almost all alternatives, including even the floating point descriptors such as SIFT and SURF.

homographies between the images. We also provide the area under the recall vs. 1-precision curve, averaged over all five image pairs in each set.

Following the test protocol employed in, e.g., [2, 3, 4, 6, 16, 24] each descriptor was extracted at image locations detected using its own original keypoint detector. Our own LATCH descriptor was applied to keypoints returned by the multi-scale Harris based detector used by the original SIFT implementation [17]. As some of the sets in the Oxford benchmark depict rotation changes and some do not, we implement rotation invariance by using the detected orientation, or the descriptors' own estimates when available.

Table 2 summarizes our results. Fig. 2 additionally provides recall vs. 1-precision curves for the Bikes and UBC sets. Aside from LDA-HASH and LDA-DIF which extract binary descriptors by using SIFT descriptors and are thus much slower, LATCH outperforms the other binary descriptors on most sets and in some cases even the much larger SIFT and SURF representations.

Learning Local Image Descriptors data-set. We next report tests on the data-set described by [5][3]. It provides a large number of detection windows along with same/not-same labels signifying whether two windows from two separate images correspond to the same physical point or not.

The test protocol used here is designed to evaluate the discriminative power of different image descriptors. Given two windows, a descriptor is extracted for each one and the distance between the two descriptors is measured. A scalar threshold is then applied to this distance in order to determine if the two descriptors are similar enough to imply that the windows should be labeled "same" or not. We use the Yosemite dataset in order to learn an optimal threshold by using linear support vector machines (SVM) [7]. Yosemite images were also used to learn patch triplet arrangements for the LATCH descriptor (Section 3.2). Testing is performed on the Liberty and Notre-Dame sets.

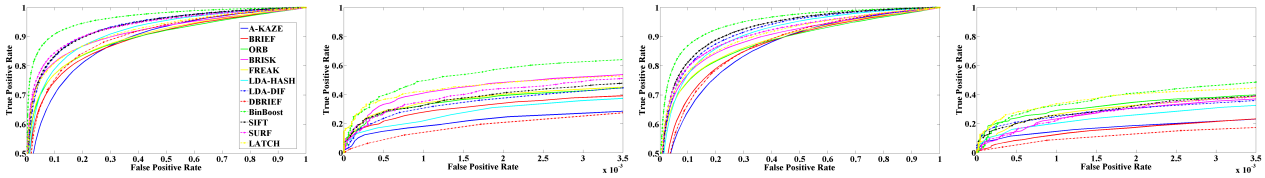
Table 3 summarizes the results in terms of accuracy, area under the ROC curve and 95% error-rate (the percent of incorrect matches obtained when 95% of the true matches are found). ROC curves for the different methods tested are presented in Fig 3. Our results show the clear advantage of the proposed LATCH descriptor over other binary descriptor designs, with LATCH outperforming the other representations, in both tests, by noticeable margins. Although BinBoost and LDA-HASH/DIF perform better than LATCH on these tests, as previously noted, this added performance comes at substantial computational costs.

Analysis: Varying descriptor size. In the tests reported above, we used a LATCH descriptor of 32 bytes. Here, we revisit the tests on the Oxford benchmark in order to evaluate the effect descriptor size has on its performance. We test varying descriptor sizes (the number of arrangements used) using 4, 8, 16, 32, and 64 bytes for the representation. Table 4 (a) summarizes our results, providing the area under the recall vs. 1-precision curve. Clearly, the performance of LATCH improves as its size grows. These results can be compared with those of Table 2.

	Notre-Dame			Liberty		
Descriptor	AUC	ACC	95% Err	AUC	ACC	95% Err
SIFT [17]	.934	.817	39.7	.928	.764	40.1
SURF [4]	.935	.866	41.1	.911	.833	55.0
LDA-HASH [26]	.916	.830	46.7	.910	.798	48.1
LDA-DIF [26]	.934	.857	38.5	.921	.836	43.1
DBRIEF [29]	.900	.830	55.1	.868	.794	61.5
BinBoost [15, 28]	.963	.907	21.6	.949	.884	29.3
SBRIEF [6]	.889	.823	63.2	.868	.798	66.7
ORB [24]	.894	.835	66.2	.882	.822	69.2
BRISK [16]	.915	.857	57.7	.897	.834	62.6
FREAK [2]	.899	.835	61.5	.887	.824	65.0
A-KAZE [3]	.885	.806	56.7	.860	.782	63.4
LATCH	.919	.855	52.0	.906	.838	56.7

Table 3. **Results on the Learning Local Descriptors dataset.** Same/not-same tests on data from [5]. Testing was performed separately on the Notre-Dame and the Liberty collections. Higher results are better for AUC and accuracy (ACC); lower results are better for the 95% error-rate (Err.). Evidently, LATCH outperforms other binary representations by clear margins.

[3] www.cs.ubc.ca/ mbrown/patchdata/patchdata.html



(a) (b) (c) (d)

Figure 3. **ROC curves for the Learning Local Descriptors data-set tests.** (a) Notre Dame set, ROC curves; (b) zoomed-in view of the low false positive region of the ROC for the Notre-Dame tests (c) ROC curves for the Liberty tests; (d) zoomed-in view of the low false positive region of the ROC for the Liberty tests.

	Descriptor	Bark	Bikes	Boat	Graffiti	Leuven	Trees	UBC	Wall	AVG
(a) Descriptor size	LATCH-4	0.012	0.282	0.013	0.038	0.269	0.014	0.131	0.030	0.099
	LATCH-8	0.029	0.353	0.028	0.074	0.319	0.039	0.167	0.080	0.136
	LATCH-16	0.053	0.394	0.044	0.098	0.355	0.064	0.192	0.133	0.167
	LATCH-32	0.065	0.415	0.057	0.119	0.374	0.082	0.215	0.175	0.188
	LATCH-64	0.073	0.425	0.070	0.131	0.381	0.097	0.239	0.205	0.203
(b) Patch size	LATCH 1×1	0.058	0.391	0.048	0.103	0.346	0.069	0.190	0.139	0.168
	LATCH 3×3	0.054	0.392	0.049	0.105	0.361	0.070	0.193	0.133	0.170
	LATCH 5×5	0.064	0.405	0.054	0.113	0.368	0.076	0.205	0.156	0.180
	LATCH 7×7	0.065	0.415	0.057	0.119	0.374	0.082	0.215	0.175	0.188
	LATCH 9×9	0.072	0.422	0.059	0.123	0.374	0.085	0.221	0.188	0.193
	LATCH 11×11	0.075	0.428	0.058	0.128	0.376	0.085	0.223	0.196	0.196
	LATCH 13×13	0.078	0.429	0.057	0.129	0.372	0.085	0.220	0.200	0.196
	LATCH 15×15	0.074	0.434	0.054	0.126	0.367	0.081	0.216	0.200	0.194
(c) Learning method	Random	0.064	0.391	0.059	0.104	0.260	0.064	0.229	0.174	0.168
	ORB/FREAK	0.073	0.396	0.066	0.108	0.267	0.074	0.239	0.187	0.176
	Proposed	0.055	0.413	0.058	0.107	0.379	0.083	0.229	0.155	0.185
	Combined	0.065	0.415	0.057	0.119	0.374	0.082	0.215	0.175	0.188

Table 4. **Analysis tests on the Oxford benchmark**. Results summarizing the performance of our LATCH descriptor using (a) different descriptor sizes (different numbers of patch triplet arrangements); (b) different patch sizes; (c) different methods of selecting arrangements. The table provides area under the recall vs. 1-precision curves. Please see text for more details.

	Notre-Dame			Liberty		
Learning Method	AUC	ACC	95% Err.	AUC	ACC	95% Err.
Random	.894	.822	57.5	.871	.793	60.6
ORB/FREAK	.902	.831	55.4	.881	.801	58.2
Proposed	.905	.842	58.6	.892	.824	61.8
Combined	.919	.855	52.0	.906	.838	56.7

Table 5. **Analysis of different learning methods on the Learning Local Descriptors dataset.** Same/not-same tests with different learning methods performed using the data from [5]. Testing was performed separately on the Notre-Dame and the Liberty collections. Higher results are better for AUC and accuracy (ACC); lower results are better for the 95% error-rate (Err.). Interestingly, our proposed learning method outperforms [2, 24]. When combining their correlated triplet elimination technique ("combined") we gain a further performance boost.

Analysis: Varying patch size. One of the key components of the LATCH descriptor is the use of pixel patches compared to sampling single pixels. We next evaluate the effect of larger pixel patches on the performance of LATCH. Here, we use a 32 byte LATCH representation, testing it with a patch sizes ranging from 3×3 to 15×15.

We report also the performance of a simpler LATCH variant, which is computed by comparing *pixel* triplets, rather than patch triplets (LATCH 1×1). Similarly to ORB, in order to handle noise pixel values are sampled following the same local smoothing. Extracting larger-patch LATCH descriptors following smoothing brought performance further down, and so we do not report these results.

Our results, summarized in Table 4 (b) demonstrate that larger patches provide more accuracy. In nearly all cases, the bigger the patches used, the higher the performance gain. The relative improvement in performance, however, decays with patches larger than 9×9 or 11×11. Here too, these results may be compared with those presented in Table 2, where LATCH was computed using 7×7 patches.

It is worthwhile to consider the performance of LATCH 1×1. Evidently, this approach almost always provides inferior results even to LATCH extracted using 3×3 patches. With the default 7×7 patches, LATCH performance is significantly better than sampling single pixels.

Analysis: Comparing learning methods. As discussed in 3.2, we propose supervised learning of optimal patch arrangements. The same approach can of course be applied to learning optimal pairs. We compare the proposed approach to that of ORB [2, 24] and also present the performance of the combined method in which the quality of the triplets is measure by their score on the "same"/"not-same" dataset, filtering out correlated triplets.

Table 4 (c) presents results on the Oxford benchamark

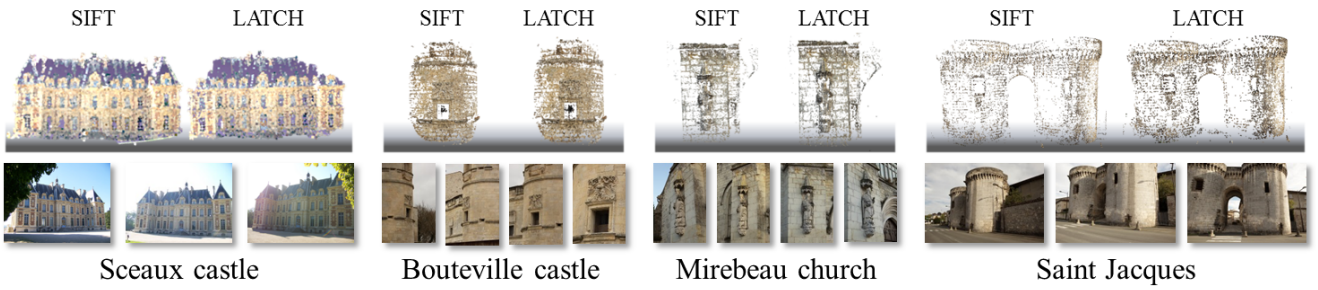


Figure 4. **Structure from motion results.** Top: 3D reconstruction results on four standard test sequences obtained with the incremental structure from motion chain method [20] using the default SIFT and our LATCH. Bottom: Input image examples from each set. Qualitatively, SIFT and LATCH provide comparable results though LATCH descriptor matching is an order of magnitude faster (Table 6).

and Table 5 on the Learning Local Descriptors set. Evidently, overall, the proposed learning method outperforms the learning method of [2, 24], and the combined method outperforms both. Unsurprisingly, random selection of patch triplets performs much worse than either of these.

4.2. Application to multi-view 3D reconstruction

One of the more challenging uses of local descriptors lies in structure from motion (SfM) applications. In order to produce accurate results, local appearances must be matched across images of the same scene, taken from possibly widely different views. Additionally, SfM methods often compare many comparisons between many interest points, and hence the efficiency of matching descriptors is also a matter of concern.

We test the use of our proposed LATCH descriptor in a SfM framework, comparing it to the SIFT descriptor often used for this purpose. To this end, we have incorporated LATCH into the OpenMVG library [20] using their incremental structure from motion chain method. We ran SfM twice, changing only the local image representations from their default SIFT to our own LATCH descriptors.

In order to isolate the effect of using LATCH rather than SIFT, both use the same keypoints, recovered by the SIFT detector implemented in the VLFeat library [30]. All OpenMVG parameters were kept at their default values apart from the ratio threshold which was 0.6 for SIFT (the default), and raised to 0.99 for LATCH (binary descriptors in general are known to be more sensitive to this value).

Sequence	SIFT	LATCH
Sceaux Castle	381.63	39.05
Bouteville Castle	4766.22	488.70
Mirebeau Church	3166.35	325.31
Saint Jacques	1651.12	169.19

Table 6. **Structure from motion descriptor matching times.** The time (seconds) required to match descriptors when producing the structure from motion results reported in Fig. 4. LATCH is consistently an order of magnitude faster to match than the standard SIFT, yet provides qualitatively similar results.

Reconstruction results for standard test image sequences [20] are provided in Fig. 4 and the time required to match the descriptors in each scene is provided in Table 6. We note that denser surfaces could conceivably be produced by running a multi-view stereo algorithm, e.g. the Patch-based Multi-View Stereo (PMVS) method of [9], following the initial reconstructions. Doing so, however, may correct errors due mismatching descriptors. We focus on the quality of the descriptors, not the final reconstruction, and so this step was not performed here.

Evidently, 3D reconstructions obtained by using both descriptors are qualitatively comparable. The time required to match our LATCH descriptors, however, is consistently an order of magnitude faster than SIFT.

5. Conclusions

Over the years, the computer vision community has invested immense efforts in a continuing effort to improve the performance of local descriptors, including the requirements they make on storage, extraction and matching time. As part of this effort, we propose a new variant to the binary descriptors representation family. Our LATCH representation enjoys the same fast matching time and small storage requirements of binary descriptors. Our tests, however, demonstrate that it outperforms other binary descriptors by wide margins, closing the gap between their performance and the performance reported by the much larger, more expensive histogram based representations.

In the future, we plan to compare LATCH with recent methods for extracting features using deep learning [35, 11]. Though these were shown to be extremely effective, their computational costs are still high, making the trade off between run time and discriminative capabilities even more acute than the one reflected by, e.g, Table 1.

A growing volume of work has shown new applications for matching all the pixels of one image to another (i.e., *dense correspondences* [13]). Such systems may benefit from efficient per-pixel representations and applying LATCH to those problems, possibly by adding scale invariance [27], is a natural next step for this line of research.

References

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *Trans. Pattern Anal. Mach. Intell.*, 28(12):2037–2041, 2006.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *Proc. Conf. Comput. Vision Pattern Recognition*, pages 510–517. IEEE, 2012.

[3] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *British Machine Vision Conf. (BMVC)*, 2013.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conf. Comput. Vision*, pages 404–417. Springer, 2006.

[5] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *Trans. Pattern Anal. Mach. Intell.*, 33(1):43–57, 2011.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European Conf. Comput. Vision*, pages 778–792. Springer, 2010.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. Conf. Comput. Vision Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[9] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, 2010.

[10] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *Proc. Int. Conf. Comput. Vision*, pages 498–505. IEEE, 2009.

[11] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015.

[12] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[13] T. Hassner and C. Liu. *Dense Image Correspondences for Computer Vision*. Springer, 2015.

[14] J. Heinly, E. Dunn, and J.-M. Frahm. Comparative evaluation of binary features. In *European Conf. Comput. Vision*, pages 759–773. Springer, 2012.

[15] V. Lepetit, T. Trzcinski, P. Fua, C. M. Christoudias, et al. Boosting binary keypoint descriptors. In *Proc. Conf. Comput. Vision Pattern Recognition*, number EPFL-CONF-186246, 2013.

[16] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proc. Int. Conf. Comput. Vision*, pages 2548–2555. IEEE, 2011.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

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

[20] P. Moulon, P. Monasse, and R. Marlet. Adaptive structure from motion with a contrario model estimation. In *Asian Conf. Comput. Vision*, pages 257–270. Springer, 2013. Available: github.com/openMVG/openMVG/.

[21] L. Nanni, A. Lumini, and S. Brahnam. Survey on lbp based texture descriptors for image classification. *Expert Systems with Applications*, 39(3):3634–3641, 2012.

[22] T. Ojala, M. Pietikäinen, and T. Mäenpää. A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. In *ICAPR*, 2001.

[23] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Proc. Int. Conf. Comput. Vision*, pages 2564–2571. IEEE, 2011.

[25] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *Trans. Pattern Anal. Mach. Intell.*, 2014.

[26] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(1):66–78, 2012.

[27] M. Tau and T. Hassner. Dense correspondences across scenes and scales. *Trans. Pattern Anal. Mach. Intell.*, 2014. To appear.

[28] T. Trzcinski, C. M. Christoudias, and V. Lepetit. Learning image descriptors with boosting. Technical report, Institute of Electrical and Electronics Engineers, 2013.

[29] T. Trzcinski and V. Lepetit. Efficient discriminative projections for compact binary descriptors. In *European Conf. Comput. Vision*, pages 228–242. Springer, 2012.

[30] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. Available: www.vlfeat.org/, 2008.

[31] L. Wolf, T. Hassner, and Y. Taigman. Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *Trans. Pattern Anal. Mach. Intell.*, 33(10):1978–1990, 2011.

[32] L. Wolf, T. Hassner, Y. Taigman, et al. Descriptor based methods in the wild. In *Proc. Conf. Comput. Vision Pattern Recognition*, 2008.

[33] X. Yang and K.-T. Cheng. Ldb: An ultra-fast feature for scalable augmented reality on mobile devices. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 49–57. IEEE, 2012.

[34] X. Yang and K.-T. Cheng. Local difference binary for ultrafast and distinctive feature description. *Trans. Pattern Anal. Mach. Intell.*, 36(1):188–194, Jan 2014.

[35] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. *arXiv preprint arXiv:1504.03641*, 2015.