Noname manuscript No.
(will be inserted by the editor)

Is there anything new to say about SIFT matching?

Fabio Bellavia · Carlo Colombo

Received: n.a. / Accepted: n.a.

Abstract SIFT is a classical hand-crafted, histogram-based descriptor that has deeply affected research on image matching for more than a decade. In this paper, a critical review of the aspects that affect SIFT matching performance is carried out, and novel descriptor design strategies are introduced and individually evaluated. These encompass quantization, binarization and hierarchical cascade filtering as means to reduce data storage and increase matching efficiency, with no significant loss of accuracy. An original contextual matching strategy based on a symmetrical variant of the usual nearest-neighbor ratio is discussed as well, that can increase the discriminative power of any descriptor. The paper then undertakes a comprehensive experimental evaluation of state-of-the-art hand-crafted and data-driven descriptors, also including the most recent deep descriptors. Comparisons are carried out according to several performance parameters, among which accuracy and space-time efficiency. Results are provided for both planar and non-planar scenes, the latter being evaluated with a new benchmark based on the concept of approximated patch overlap. Experimental evidence shows that, despite their age, SIFT and other hand-crafted descriptors, once enhanced through the proposed strategies, are ready to meet the future image matching challenges. We also believe that the lessons learned from this work will inspire the design of better hand-crafted and data-driven descriptors.

Authors
Fabio Bellavia is with the Department of Mathematics and Computer Science, Università degli Studi di Palermo, Via Archirafi, 34, 90123 Palermo, Italy. E-mail: fabio.bellavia@unipa.it
Carlo Colombo is with the Department of Information Engineering, Università degli Studi di Firenze, Via di S. Marta, 3, 50139 Firenze, Italy. E-mail: carlo.colombo@unifi.it

Keywords SIFT · sGLOH2 · quantization · binary descriptors · symmetric matching · hierarchical cascade filtering · deep descriptors · keypoint patch orientation · approximated overlap error

1 Introduction

Image matching through local descriptors is the core of many computer vision applications including, among others, object detection [40] and recognition [70], image stitching [14], three-dimensional reconstruction [54] and visual odometry [23]. This justifies the huge efforts and progresses made over the years on this topic.

Local descriptors are conventionally divided into hand-crafted and data-driven [20] according to the process used for extracting the descriptor vector from the local keypoint patch. Hand-crafted descriptors mainly employ histograms to accumulate a chosen local patch property. The property upon which the popular Scale Invariant Feature Transform (SIFT) descriptor [40] and its inexhaustible crowd of descendants [3,9,19,22,26,31,48] are based is gradient orientation. Other histogram-based descriptors use pixel ordering [65], Haar wavelets [8], convolutions with Gaussian [58] or other kernels [50], and binary pixel comparisons [28]. Data-driven descriptors are those whose structure and design have been tuned and refined according to training data. Training is used either to obtain low-dimensional binary descriptors [20,55,61], or to find the best setup in the descriptor parameter space [7, 15, 20]. Recently, the so called deep descriptors [37,45,53,57,66] have appeared, leveraging deep learning and the growing availability of large training datasets [15, 46].

The main parameters that characterize descriptor performance are discriminability, memory storage and

matching speed. While the ideal descriptor should conjointly optimize such parameters, real descriptors are usually the result of a trade-off, where a higher discriminative power is balanced by a larger size and slower computations, and vice versa. This can be referred to as "discriminability vs space-time dilemma."

This paper addresses the problem of enhancing descriptor matching performance in two complementary ways. On the one hand, we show that, for SIFT and other SIFT-like descriptors, length and matching times can be significantly compressed through a proper quantization scheme. On the other hand, we explain how the discriminative power of *any* (hand-crafted or data-driven) descriptor can be increased by a carefully designed contextual matching approach. An important consequence of the above is that, for what concerns the design of SIFT-like descriptors, one can avoid the discriminability vs space-time dilemma above, and enhance conjointly all performance parameters. The proposed quantization solution to space-time compression will prove to be preferable over the more common descriptor binarization, that nevertheless remains of some theoretical interest, and will be addressed in the paper.

The paper also presents the results of a thorough comparative evaluation, carried out on both planar and non-planar scenes, and highlighting the impact of the proposed descriptor enhancements on the recent state-of-the-art. Experimental evidence shows that SIFT and other hand-crafted descriptors, once suitably enhanced, exhibit very interesting properties, which make them still appealing for image matching. Results also show that the usual SIFT patch orientation assignment strategy, employed by most descriptors, is not optimal, and can be conveniently replaced by more recent alternatives based on deep learning [69]. Additionally, it is also discovered that quantizing descriptors and matching them using the L_1 distance in the place of the standard L_2 distance yields the same computational efficiency of binary descriptors and the same high accuracy of unquantized descriptors. Finally, according to our experiments, deep descriptors do depend heavily on the training set and hardware setup, which may limit their use in general contexts and applications.

The paper is organized as follows. Section 2 introduces the novel quantization scheme for SIFT-like descriptors. In the particular case of SIFT, quantization shrinks each byte either into nibbles (4 bits) or into packets of 3 bits, according to application requirements. The resulting Packed SIFT (PSIFT) reduces SIFT from the original 128 bytes down to 48 bytes, thus at least halving the storage requirements and reducing the memory bandwidth, yet without compromising running times and matching accuracy. The same quantization scheme is also applied to the recent doubled shifting Gradient Histogram of Orientations (sGLOH2) [9], with similar results. Section 3 expounds Binary SIFT (BiSIFT), a descriptor generated according to the general approach developed in [9]. BiSIFT is 61 bytes and uses the Hamming distance for matching. Experimental evidence is provided of the limitations of binary descriptors based on sorting against those based on binary comparisons. Section 4 presents Hierarchical Cascade Filtering (HCF) as a strategy to supersede the descriptor matching process and reduce the matching time, thanks to the introduction of small descriptor fingerprints to pre-filter the data. HCF is shown to be very useful in the case of `float` descriptors, or when the L_2 distance is employed for matching. The advantages of using a symmetric Nearest Neighbor Ratio (sNNR) for matching are also investigated: To the best of our knowledge, such analysis has never been carried out before. Section 5 includes an exhaustive evaluation of the proposed enhancements, that were embedded into and also compared against recent state-of-the-art approaches. Experimental results are presented and discussed for planar and non-planar scenes. The latter were obtained with a novel evaluation benchmark, designed around a suitable approximation of the patch overlap error [11]. The evaluation also includes an analysis of the influence of patch orientation assignment on the matching process. Final considerations are provided and future work is outlined in Sec. 6.

2 Quantization

2.1 Related work

SIFT is probably the most renown among the histogram-based descriptors. It is constructed as the concatenation of gradient orientation histograms for different keypoint patch areas. Over the years, SIFT has been deeply analyzed, and constantly ameliorated. In particular, alternative grid arrangements and pooling schemes were proposed in order to get rotational invariance [22, 33] or improve robustness [18, 19]. Dimensionality reduction [26, 31] and alternative matching distances [3, 39] were also introduced so as to increase discriminability. The problem of space-time compression has been traditionally addressed through data-driven techniques such as dimensionality reduction [29, 31], hashing [55], binarization [4, 9, 20] and quantization [17, 63, 67].

With the noticeable exception of [67], where it was analyzed separately from other factors in descriptor design, descriptor vector quantization was somewhat underestimated by the recent literature, and considered

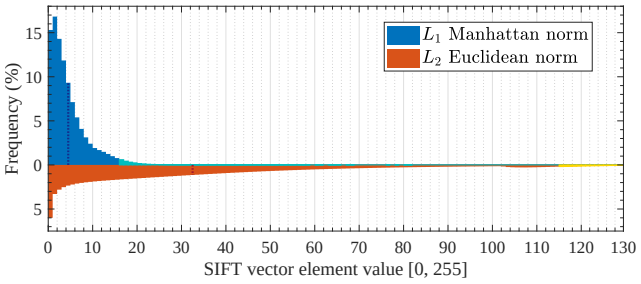


Fig. 1 Distribution of the quantized SIFT vector element values after normalization by L_1 and L_2. Dashed lines indicate mean values. Values outside the 98% of probability are shown in light blue and yellow. (Best viewed in color.)

as a merely secondary optimization technicality. Nevertheless, it is sensible to suppose that the scalability and efficiency of practical applications strongly rely on quantization for reducing the computational complexity of the matching process in terms of running time, storage requirements, and memory bandwidth.

2.2 Packed SIFT

Our quantization scheme originates from the observation that, when matching two histograms, the Euclidean L_2 distance tends to emphasize large errors occurring on a few bins with respect to small errors on the remaining majority of bins. This behavior can be mitigated by employing alternative distances for histogram comparison, such as the lower order Manhattan L_1 [9] or Hellinger's [3] distances.

Let $\mathbf{x}$ be the real-valued SIFT vector obtained after the last normalization by the L_2 norm:

$$\mathbf{x} = [x_1, \ldots, x_{l_s}] \tag{1}$$

with $x_i \geq 0$, $l_s = 128$ and $\sum_i x_i^2 = 1$. The final SIFT vector is obtained by quantizing each x_i into the byte range $[0, 255]$ as

$$q_S(x_i) = \min\left(\lfloor m\, x_i \rfloor, 255\right) \tag{2}$$

where $\lfloor \cdot \rfloor$ is the floor function and $m = 512$.

RootSIFT is a modern variant of SIFT that is obtained by first normalizing $\mathbf{x}$ by the L_1 norm

$$y_i = \frac{x_i}{\sum_j x_j} \approx \frac{q_S(x_i)}{\sum_j q_S(x_j)} \tag{3}$$

and then extracting the square root of each y_i. Computing the Hellinger's distance between two SIFT vectors is equivalent to computing the L_2 distance between the corresponding RootSIFT vectors [3].

Figure 1 shows the distribution of the quantized vector values $\dot{y}_i = q_S(y_i)$ and $\dot{x}_i = q_S(x_i)$, obtained from the SIFT descriptors of the Oxford dataset [44] after normalization by L_1 and L_2, respectively. The distribution of $\dot{y}_i$ is less dispersed (with mean value, standard deviation and 98% cutoff value equal to $\mu_y = 3.99$, $\sigma_y = 4.08$ and $y^\star = 16$, respectively) than that of $\dot{x}_i$ ($\mu_x = 32.52$, $\sigma_x = 30.95$ and $x^\star = 115$). The above discrepancy between distributions suggested us to devise the following quantization scheme, which is alternative to the one of Eq. 2 and based on the L_1 norm:

$$q_{g,t}(y_i) = \min\left(\left\lfloor \frac{g(m\, y_i)}{g^*}\, 2^t \right\rfloor, 2^t - 1\right) \tag{4}$$

In Eq. 4, $g : \mathbb{R} \to \mathbb{R}$ is a monotonic function, g^* is a scale factor depending on g, $\lfloor \cdot \rfloor$ denotes the rounding operation and t is the saturation threshold expressed in bits. Three possible g maps are considered hereafter:

$$I(y_i) = y_i \tag{5}$$

$$R(y_i) = y_i^{1/2} \tag{6}$$

$$N(y_i) = \begin{cases} y_i & \text{if} \quad y_i < \bar{y} \\ (y_i - \bar{y})^{1/2} + \bar{y} & \text{otherwise} \end{cases} \tag{7}$$

where $\bar{y} = \mu_y - 1$. The scale factors replacing g^* in Eq. 4 are set to $I^* = I(y^\star)$, $R^* = R(y^\star)$ and $N^* = N(y^\star - 1) + 1$, respectively. The corresponding quantization maps are plotted in Fig. 2 for $t = 3$ (a) and $t = 4$ (b). Notice that $N(y_i)$ behaves linearly as $I(y_i)$ for values lower than μ_y, and according to the square root $R(y_i)$ for higher values. The function N can be viewed as a lower order analogue of the Huber loss function, that replaces the squared/linear components of the Huber loss respectively with linear/root square components. For the sake of completeness, we show that the theoretical expected value $E(\dot{y}_i)$ can be used in the place of the empirical mean $\mu_y = 3.99$, under the assumption that the distribution of $\dot{y}_i$ is independent of the bin position. Indeed, since $l_s\, E(\dot{y}_i) = \sum_i E(\dot{y}_i) = E\left(\sum_i \dot{y}_i\right) \approx E\left(m \sum_i y_i\right) \approx E(m) = m$, it holds

$$E(\dot{y}_i) \approx \frac{m}{l_s} = \frac{512}{128} = 4 \tag{8}$$

The effects of packing by quantization the descriptor vector elements are illustrated in Fig. 3. When the saturation threshold is $t = 4$ bits, half byte (one nibble) can be used to store a single vector value. In the case of SIFT, descriptor length is halved from 128 to 64 bytes, and so is the memory storage. Pairs of consecutive nibbles can be stored exactly into a single byte, without waste of space or memory alignment issues. Nibble packing and unpacking can be done efficiently by shifting and masking bits as described in the left part of Fig. 3. If the total time for retrieving and unpacking nibble vectors is not greater than that required

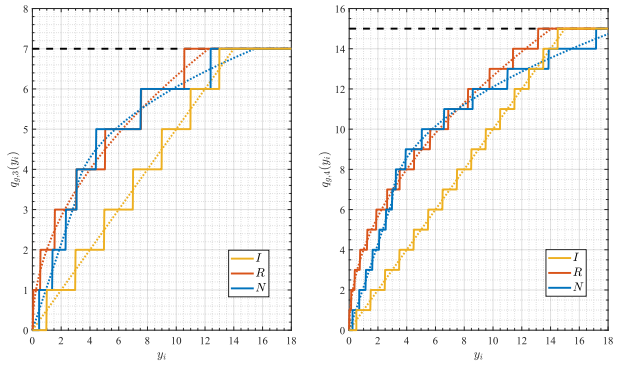


Fig. 2 Quantization maps $q_{g,t}(y_i)$ in the case of (a) packets of 3 bits ($t = 3$) and (b) nibbles ($t = 4$). Dashed lines refer to the values before rounding off and saturation. (Best viewed in color.)

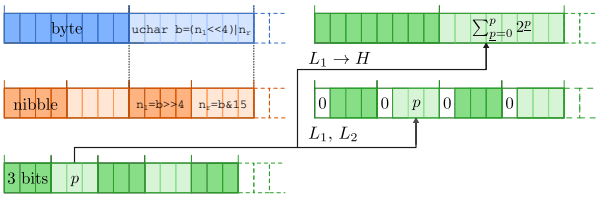


Fig. 3 (left) Memory storage alignments for the proposed quantized descriptors and (right) corresponding stretching for vector manipulation in the case of packets of 3 bits. Bit operations are shown in C code. (Best viewed in color.)

to retrieve the equivalent byte vectors, then all data operations with packed SIFT and $t = 4$ can be carried out in-place, i.e., without unpacking the vector into memory before computation. When $t = 3$ bits, only 48 bytes (i.e., 3/8 of the original SIFT length) are needed for storage purposes. However, descriptors have to be unpacked before any vector manipulation, due to memory alignment issues—see the right part of Fig. 3. The most trivial unpacking strategy is to interleave a 0 bit between values, thus leading back to the nibble case. Alternatively, if descriptor comparisons have to be done using the L_1 distance, each i-th packet with value $q_{g,3}(y_i)$ can be stretched into a byte so that the first $q_{g,3}(y_i)+1$ bits are set to 1 and the remaining to 0. As the Hamming distance H between two of these unpacked vectors is the same as the L_1 distance between the original vectors, this representation"trick" can save matching time on all hardware configurations where H is faster than L_1 to compute.

2.3 Experimental results

Table 1 reports on image matching results obtained with the proposed quantization scheme with planar scenes from the WISW [10] and Oxford [44] datasets (19 sequences in all, 15 from WISW and 8 from Oxford, 4 sequences being in common between the datasets. Each sequence contains 6 images, the first of which is used as reference, for a total of $19 \times (6-1) = 95$ image pairs) which include viewpoint changes, the most challenging sources of image distortion, also in combinations with other relevant image transformations, such as illumination changes. Matching accuracy is expressed in terms of mean Average Precision (mAP), based on the overlap error. The setup is the same of [9].

Besides SIFT (the VL SIFT implementation [64]), the recent SIFT-like descriptor sGLOH2 [9] using the sGOr2a* matching strategy is also tested (the distribution of the sGLOH2 values is almost identical to the one shown in Fig. 1 for SIFT, see additional material). Correspondences were obtained using Nearest Neighbor Ratio (NNR) matching, with the descriptor default distance (L_2 for SIFT, permutation-based L_1 for sGLOH2). Results for SIFT matched with the L_1 distance (referred to as SIFT$_{L_1}$) are reported as well. In the table, results with the original quantized descriptor implementations, i.e. $\dot{x}_i$ for SIFT and SIFT$_{L_1}$, and $\dot{y}_i$ for sGLOH2, are reported in the first column. The second column shows the results when the mapping function g is applied without quantization, thus obtaining floating-point descriptor vectors (in particular, $I(y_i)$ corresponds to real-valued descriptor vectors normalized by L_1). The square root R gives the best results on unquantized descriptors, confirming previous evidence [3] (notice that $R(y_i)$ in the SIFT row corresponds to RootSIFT).

The table clearly shows that quantization of a given descriptor does not affect significantly its matching accuracy. Better still, 3 bits quantization with N mapping $q_{N,3}(y_i)$ slightly improves accuracy, yielding the best results for both SIFT and sGLOH2. Similarly to PCA, $q_{N,3}(y_i)$ appears to reduce the effects of noise on data, but differently from PCA it does not operate on vector dimensions but on value ranges, thus producing discrete values that can be represented as integers and allow for efficient computation.

Table 2 reports the average running times required for matching two descriptors on an Intel Core i7-3770K with 8 GB of RAM and Ubuntu 16.04. Two different versions of the code were implemented. In the first implementation, the code was optimized explicitly using SSE 4.1 instructions on 128 bits XMM registers, and the `popcount` (64 bits) instruction for an efficient Hamming distance. In the second implementation the code was compiled with no explicit optimization[1], save for `popcount` (32 bits), so as to generate a more portable

[1] only the "`-O3`" flag was enabled in GCC.

Table 1 Quantized descriptors mAP (%) on planar scenes. For each row, the 1^{st}, 2^{nd} and 3^{th} best values are colored respectively in red, blue and green. (Best viewed in color.)

	$\dot{x}_i \vert \dot{y}_i$	$g(y_i)$			$q_{g,3}(y_i)$			$q_{g,4}(y_i)$		
		I	R	N	I	R	N	I	R	N
SIFT	63.60	62.94	63.87	63.82	62.78	62.46	64.01	63.13	62.70	62.84
SIFT$_{L_1}$	63.97	63.58	63.75	63.81	62.92	62.42	63.74	63.52	62.75	63.19
sGLOH2	77.20	77.31	78.29	75.54	77.05	77.25	78.36	76.73	77.76	76.79
bits/element	8	32	32	32	3	3	3	4	4	4

Table 2 Matching average running times (ns) for quantized descriptors. 3 bits packets are stretched to one byte and matched with the Hamming distance.

			`uchar`	`float`	nibble	3 bits
single thread	SSE 4.1	SIFT	37	44	34	–
		SIFT$_{L_1}$	12	38	16	20
		sGLOH2	418	667	414	441
	popcount	SIFT	67	157	126	–
		SIFT$_{L_1}$	134	163	144	47
		sGLOH2	1110	1396	1070	500
multi thread	SSE 4.1	SIFT	12	15	11	–
		SIFT$_{L_1}$	5	15	7	6
		sGLOH2	124	180	117	120
	popcount	SIFT	19	43	34	–
		SIFT$_{L_1}$	37	49	41	13
		sGLOH2	288	353	293	132

and manageable software. Running times are provided for different descriptor value allocation units (notice that C types `uchar` and `float` are respectively 1 and 8 bytes on `x64` CPU registers, and 1 and 4 bytes on XMM registers).

Results are given for both single-threaded and multi-threaded (8 cores) code execution. Multi-threading scales down the running times without changing the relative time performance obtained with single-threading. Hence, the following discussion will be limited to single-threaded code efficiency.

With SSE 4.1 optimization, the best performance of all is achieved by matching `uchar` SIFT using the L_1 distance, that can be computed with just one instruction, i.e., the Sum of Absolute Values (SAD). 3 bits SIFT$_{L_1}$ with 8 bits stretching (see Sec. 2.2) takes almost twice the time, as two instructions are required (i.e., the bitwise `xor` followed by `popcount`) to compute the Hamming distance. Matching nibble SIFT$_{L_1}$ in place as explained in Sec. 2.2 is slightly slower than using `uchar` SIFT$_{L_1}$, yet much faster than `float` SIFT$_{L_1}$. Nibble SIFT$_{L_1}$ can thus be conveniently used instead of `uchar` SIFT$_{L_1}$ in the presence of memory bandwidth constraints. Conversely, nibble SIFT$_{L_1}$ can always be converted into `uchar` SIFT$_{L_1}$, thus enjoying a faster distance computation, at the expense of a larger memory storage[2]. Notice that `float` SIFT$_{L_1}$ is about four times slower than `uchar` SIFT$_{L_1}$, as the size of a `float` is four times that of a `uchar` on XMM registers. Similar considerations could be derived for other data types, such as `ushort` (2 bytes) and `uint` (4 bytes): Running times are roughly proportional to data length. Concerning matching SIFT with the default L_2 distance, there is virtually no gain in using `uchar` in the place of `float`: There is in fact no equivalent to SAD for L_2. Differently from the L_1 case, nibble SIFT is faster than `uchar` SIFT. This is probably due to the fact that nibble integers raised to the square can always be represented with a `uchar` ($(2^4-1)^2 \leq 255$), while `uchar` integers cannot. Comparing the time performance figures of `uchar` SIFT$_{L_1}$ and classical SIFT, we notice that the former is always between three and four times faster the latter, depending on the data type used (`float` is the slowest, nibble is the fastest). Finally, sGLOH2 has the worst running times, due to the more complex matching distance employed.

With no explicit SIMD vectorization, 3 bits SIFT$_{L_1}$ with 8 bits stretching gives the best results, followed by `uchar` SIFT$_{L_1}$, for which the compiler is able to automatically optimize the code. The other data types behave quite similarly to each other.

According to the results obtained, the $q_{N,3}$ quantization scheme is the one with the best overall performance in terms of descriptor size, accuracy, computational efficiency and adaptability to coding constraints. From now on, the SIFT descriptor quantized according to $q_{N,3}$ will be referred to as PSIFT.

A final word must be said about computing matching distances using GPUs. Currently, only L_2 is implemented through optimized `float` matrix multiplication. Due to the huge bandwidth reduction obtained by quantizing `float` into nibble (about 1/8), and considering that GPUs have instructions that are equivalent to those of SSE 4.1 (e.g., `_vsadu4` and `_popcll` are the

[2] More generally, any smaller data type can be accommodated, at matching time, into larger data types, but not viceversa. For this reason, $q_{N,3}(y_i)$ can be used with any of the data types examined, while RootSIFT, being a `float` descriptor, cannot even be put into `uchar`.

CUDA GPU equivalent instructions of `_mm_sad_epu8` and `__builtin_popcountll`, that respectively were used to implement SAD and `popcount` in our CPU optimized code), it would be worth investigating alternative implementations of both L_2 and L_1 for smaller data types.

3 Binarization

3.1 Related work

Binary descriptors are designed to be compact and fast in both extraction and matching, at the expense of their robustness. They are defined as concatenations of binary values, to be matched using the Hamming distance. Descriptor binary values are usually obtained from intensity comparisons among regions of the same keypoint patch [2, 16, 28]. However, alternative characterizations exist for binary descriptors. For instance, a non binary descriptor can be binarized by thresholding the result of manipulations inside the descriptor vector space [55, 61] or, in the case of histogram-based descriptors, by relying on comparisons among histogram bins [4, 9]. Existing binary descriptors based on patch intensity mainly differ from each other by the size and location of the patch regions under comparison, that can either be chosen according to some pre-defined pattern [2,28], or learned and optimized according to patch data [16, 20]. Patch intensity alterations due to specific geometric deformations of the patch (e.g., scale changes or more general affine transformations) can also be exploited so as to improve descriptor matching robustness [7, 68].

In [4] a binary version of SIFT was proposed, named BInarization of Gradient Orientation Histograms (BIGOH), concatenating binary comparisons between successive histograms bins for each grid cell (i.e., between SIFT descriptor vector elements). BisGLOH2 [9] binarizes sGLOH2 by comparing all possible pairs of histograms bins in a hierarchical way. This kind of approach differs from those based on intensity or gradient comparisons, since it works directly on the final descriptor, i.e., on histograms (that are more robust) and at different patch levels. Additionally, it does not rely on information about descriptor space provided by training data, as data-driven binary descriptors based on hashing, linear embedding or thresholds do. Hereafter, a new binary SIFT derived from the BisGLOH2 approach is discussed and evaluated.

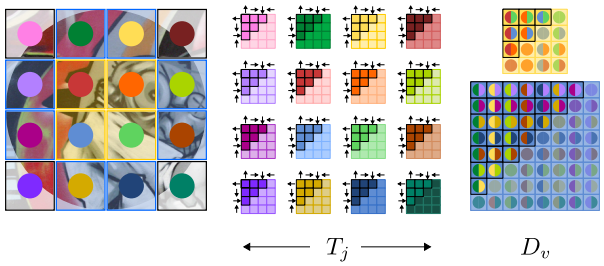


Fig. 4 *Left*: the SIFT patch is subdivided into 4×4 cells wherein gradient orientation histograms h_j are computed (the highlighted circular area stands for the Gaussian mask used for histogram weighting). Dots and background colors indicate respectively the cell and its label (blue corresponds to $v = 1$, yellow to $v = 2$, see text). *Middle*: T_j strings are defined according to the comparison of the histogram orientation bins inside each cell. Dropped comparisons are indicated with shallow colors. *Right*: D_v strings are obtained similarly. In this example, $d = 4$ orientations are used. (Best viewed in color.)

3.2 Binary SIFT

BiSIFT compresses the original SIFT into less than half of its size (explicitly, 61 bytes), like PSIFT. Nevertheless, while PSIFT is 128 bytes long when matched using the Hamming distance, BiSIFT matching requires 61 bytes only.

The BiSIFT descriptor is defined as follows. The real-valued SIFT descriptor vector $\mathbf{x}$ of Eq. 1 is normalized by L_1 norm, and then quantized to $\bar{m} = 2048$ levels, i.e., $z_i = \lfloor \bar{m} y_i \rfloor$, where y_i is defined in Eq. 3. The vector $\mathbf{z}$ is made up of $r = 16$ gradient orientation histograms h_j, $j = 1, \ldots, r$, corresponding to the 4×4 Cartesian grid cells into which the SIFT patch is subdivided (see Fig. 4, *left*). Each histogram h_j is in turn given by the concatenation $\oplus$ of $d = 8$ orientation bins $h_{j,w}$, $w = 1, \ldots, d$, so that $\mathbf{z} = [z_1, \ldots z_{l_s}]$ can be expressed as

$$\mathbf{z} = \bigoplus_{j,w} h_{j,w} \tag{9}$$

The BiSIFT descriptor $\mathbf{b}$ is defined as

$$\mathbf{b} = \bigoplus_{j} T_j \bigoplus_{v} D_v \tag{10}$$

where the binary strings T_j and D_v are obtained from a linear re-arrangement of upper triangular matrices defined in terms of the components $h_{j,w}$ of $\mathbf{z}$, and $v \in \{1, 2\}$ (see Fig. 4, *middle*, *right*). Explicitly,

$$T_j = \bigoplus_{\substack{(w_1, w_2) \\ w_1 < w_2}} (h_{j,w_1} \leq h_{j,w_2}) \tag{11}$$

where w_1, w_2 scan the same index range of w. Each string T_j has length $l_t = \frac{d(d-1)}{2}$, and its bits are obtained from pairwise comparisons of the bins $h_{j,w}$ in-

side the histogram h_j. Likewise, assuming different labels $\mathcal{L}_j = v$ for each grid cell of the SIFT patch (see Fig. 4, *left*), the binary strings D_v are defined as

$$D_v = \bigoplus_{\substack{(j_1,j_2)\\ j_1<j_2\\ \mathcal{L}_{j_1}=\mathcal{L}_{j_2}=v}} \left(\sum_w h_{j_1,w} \leq \sum_w h_{j_2,w} \right) \tag{12}$$

where j_1, j_2 scan the same index range of j so that only index pairs with the same label value v contribute to D_v. Since the grid cells labeled 1, 2 are respectively $c_1 = 8$ and $c_2 = 4$, the total length l_b in bits of the BiSIFT descriptor is

$$l_b = r\, l_t + \frac{c_1(c_1-1)}{2} + \frac{c_2(c_2-1)}{2} = 482 \tag{13}$$

and BiSIFT can be efficiently stored into $\lceil 482/8 \rceil = 61$ bytes. The main difference between BiSIFT and BisGLOH2 is in the different grid arrangements, Cartesian for BiSIFT and circular for BisGLOH2, which implies a different computation of D_v, and the fact that while BisGLOH2 is rotationally invariant, BiSIFT is not so.

Descriptor matching with BiSIFT is done, similarly to BisGLOH2, using the modified Hamming distance H_S, weighing twice the strings D_v with respect to the strings T_j:

$$H_S(\mathbf{b}, \bar{\mathbf{b}}) = \sum_{i'=1}^{r\, l_t} \texttt{xor}(b_{i'}, \bar{b}_{i'}) + 2 \sum_{i''=r\, l_t+1}^{l_b} \texttt{xor}(b_{i''}, \bar{b}_{i''}) \tag{14}$$

This can be computed efficiently on standard hardware, since $r\, l_t = 448$ is a multiple of 64, so that no memory alignment issues arise.

Notice that the discriminability power of BiSIFT decreases with the number of bins of histogram h_j having duplicate values. This phenomenon is illustrated in Fig. 5, comparing the case when all the bin values are distinct from each other (*first row*, *left*) with the case of two bin pairs with equal values (*first row*, *right*). In the former case, the upper and lower extra-diagonal triangular parts of the $d \times d$ matrix Q_j embedding the results of all possible pairwise bin comparisons have complementary binary values (*second row*, *left*). This does not hold in the latter case (*second row*, *right*). Therefore, the entries of the string T_j, which correspond to the upper extra-diagonal triangular part of Q_j, contain all possible bin comparison information in the former case, but not in the latter case. Analogous considerations can be made for D_v.

The observation above has important consequences on the possibility of further compressing BiSIFT. Indeed, assuming for the moment that there are no duplicate values inside h_j, the string T_j can be mapped

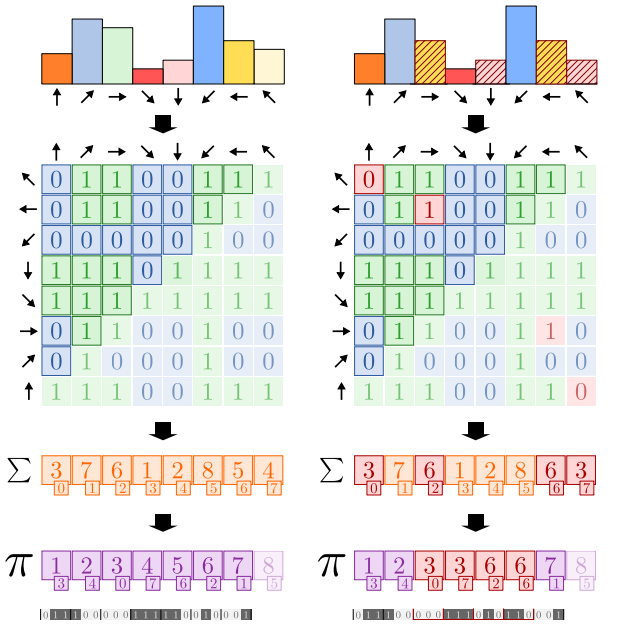


Fig. 5 BiSIFT compression (see text). The histogram h_j (*first row*), in the case of distinct (*left*) or duplicate (*right*) values for strings T_j. *Second row*: the matrix Q_j, whose upper triangular part (solid color) is encoded in T_j. *Third and fourth rows*: column-wise sums of Q_j, before and after sorting. Indexes w are reported as subscripts. *Fifth row*: the final encoding for Bi$^\pi$SIFT. Values giving rise to representation ambiguities are marked in red. In this example $d = 8$. (Best viewed in color.)

uniquely into the sorted list

$$h_{j,\pi_1} < h_{j,\pi_2} < \ldots < h_{j,\pi_d} \tag{15}$$

for some permutation π_w of the index w. From Fig. 5 (*third and fourth rows*, *left*), it is easy to see that π_w can be recovered from Q_j (hence, from T_j) by sorting the array of its column-wise sums. Using $\log_2(\lceil d-1 \rceil) = 3$ bits for coding the first $d-1$ permutation indexes π_w (the last one is unnecessary, as it is unequivocally determined given the others), only $(d-1)\log_2(\lceil d-1 \rceil) = 21$ bits are needed to store a representation of T_j, instead of the $l_t = 28$ bits required for the uncompressed string. As similar considerations hold for D_v, a more compact descriptor Bi$^\pi$SIFT of 46 bytes only can be defined, to be unpacked in the matching process. In the case of duplicate bin values, occurring when the "=" sign applies in Eq. 11, ambiguities arise during sorting (see From Fig. 5, *third and fourth rows*, *right*). These can be removed through stable sorting, i.e., if $w_1 < w_2$ and $h_{j,w_1} = hj, w_2$ at packing time, then $h_{j,w_1} < h_{j,w_2}$ at unpacking time. It has to be eventually remarked that BiSIFT and unpacked Bi$^\pi$SIFT are identical only in the case of distinct bin values inside the same histogram. Otherwise, the permutation-based compression of BiSIFT into Bi$^\pi$SIFT is lossy, thus yielding a possibly less discriminant descriptor.

Table 3 Binarized descriptors mAP (%) on planar scenes, average matching running times (ns) and byte length.

	mAP (%)	SSE 4.1		`popcount`		byte length$^{\triangleleft}$
		⊡	⊞	⊡	⊞	
BiSIFT	62.10	13	5	17	6	$^{61}/_{61}$
Bi$^{\pi}$SIFT	57.93	13	4	17	6	$^{46}/_{61}$
BisGLOH2	76.19	377	90	401	98	$^{126}/_{160}$
Bi$^{\pi}$sGLOH2	66.16	356	87	390	93	$^{95}/_{160}$

⊡/⊞ single/multi thread running times (ns)
$\triangleleft$ packed/unpacked

3.3 Experimental results

Table 3 reports results in terms of mAP and average running time for matching several binary descriptors derived from SIFT and sGLOH2. The matching setup of Sec. 2.3 was used. The "Bi" and "Bi$^{\pi}$" prefixes indicate uncompressed and compressed binarization, respectively.

Concerning matching accuracy, Bi-descriptors exhibit slightly inferior (about 2-4%) mAP values with respect to their quantized counterparts (compare with Table 1), while for Bi$^{\pi}$-descriptors this degradation is more evident (up to 15% of mAP difference).

Concerning running times, there is no noticeable difference between the two alternative binarizations. In the case of SSE 4.1 optimization, while BiSIFT performs nearly the same as PSIFT, BisGLOH2 exhibits an improvement with respect to its quantized counterpart (compare with Table 2). When no explicit optimization except for the `popcount` instruction is available, binarized descriptors have a clear advantage over the quantized ones, especially in conjunction with multithreading. This is sensible, since PSIFT length is more than twice the corresponding Bi-descriptor counterpart, while both use the Hamming distance.

As already noted by other authors [72], binary descriptors matched with the Hamming distance are almost equivalent to descriptors based on sorting matched with the Kendall τ correlation function. However, from the results above, Bi$^{\pi}$-descriptors, that rely more on sorting than on comparing, do not seem to be valid alternatives to Bi-descriptors, since their more compact storage size does not justify their major gap in terms of matching accuracy. This suggests that just sorting is less informative than having a partial list of comparisons, because the probability of obtaining equal bin values with discretized image values is relevant.

4 Contextual matching

4.1 Related work

Besides defining a measure of (dis)similarity between descriptors, a good criterion for matching must operate *globally* so as to adapt itself to data. Most matching criteria are based on the concept of Nearest Neighbor (NN) search. In particular, NN and NN Ratio (NNR) are the most popular and plain matching strategies [44], upon which more complex variants have been built. RANdom SAmple Consensus (RANSAC) [24], spatial geometrical constraints [12, 34, 42], cross-checking and relaxation of correspondences [71] are some of the techniques employed to achieve matching robustness. Among similarity measures, the Euclidean, Manhattan and Hamming distances still remain the most common choices. This is mainly due to their ease of implementation, computational efficiency on current hardware, and adaptability to large-scale NN search problems through Approximated NN search (ANN) [30, 43, 49] or cascade filtering [2]. Several cascade filtering approaches have been designed for speeding-up SIFT. Some of these are not adaptive and require off-line preprocessing [59], while others do not explicitly take into account the inter-relations among the descriptor histogram bins [62].

In the following, a novel cascade filtering strategy for SIFT-like descriptors, both adaptive and hierarchical, is introduced. An effective matching strategy based on NNR is also presented and discussed.

4.2 Hierarchical Cascade Filtering

In the design of BiSIFT in Sec. 3, the descriptor is subdivided into the D_v and T_j strings, representing respectively its coarser and finer scale levels. This idea inspired us to design a hierarchical framework for contextual matching, where patch cells represent the coarser scale level, and orientation bins the finer level. In our interpretation of contextual matching, the distance between two corresponding descriptors is statistically lower than the average distance between two non-corresponding descriptors. Hierarchical Cascade Filtering (HCF) between two sets of descriptor vectors works as follows. Short fingerprints of all descriptors are first computed at the coarsest scale. Unlikely matches are then removed using a statistical criterion. Finally, the remaining candidate matches are processed as usual, thus avoiding unnecessary computation of distances between unlikely matches. HCF implementation is discussed hereafter only in the case of exhaustive brute force matching, that should be preferred, when possible, to ANN approaches. Nevertheless, thanks to its hi-

erarchical partitioning scheme, HCF can be adapted to other NN search approaches as well.

The SIFT fingerprint $\dot{\mathbf{f}}$ has length $r = 16$ elements. Using the notation of Sec. 3, each element $\dot{f}_j$ is defined as

$$\dot{f}_j = \sum_w h_{j,w} \tag{16}$$

where summation extends over the bin orientation index. The sGLOH2 fingerprint $\ddot{\mathbf{f}}$ has length 32, and enjoys the additional property of being rotationally invariant. It is defined as the two-rings version of the Rotation Invariant Feature Transform (RIFT) descriptor [33], obtained by summing histogram values over the region direction index introduced in [9]. The fingerprint for binarized descriptors is $\dddot{\mathbf{f}} = D_v$, defined in Eq. 12. Fingerprints for non-binary descriptors can be efficiently computed and stored: This needs to be done only once at runtime. No fingerprint needs to be computed for a binary descriptor, as it is simply a substring of the descriptor itself. Notice that the data type of a fingerprint depends on the data type of the corresponding descriptor. In particular, `uchar`, `ushort` and `float` fingerprints are required for nibble, `uchar` and `float` descriptors, respectively (e.g., for nibbles, the maximum fingerprint value is $(2^4 - 1) \times d = 120 < 255$, with $d = 8$).

HFC matching of two descriptor sets $S_1 = \{\mathbf{d}_1^1, \ldots, \mathbf{d}_{s_1}^1\}$ and $S_2 = \{\mathbf{d}_1^2, \ldots, \mathbf{d}_{s_2}^2\}$ removes iteratively unlikely matches according to the fingerprint. At the n-th iteration, the (k_1, k_2) entry of the distance matrix $M^n \in \mathbb{R}_{\geq 0}^{s_1 \times s_2}$ is computed as

$$M_{k_1,k_2}^n = \begin{cases} M_{k_1,k_2}^{n-1} \text{ if } M_{k_1,k_2}^{n-1} \leq \mu_{k_1}^{n-1} \text{ and } M_{k_1,k_2}^{n-1} \leq \mu_{k_2}^{n-1} \\ \infty \qquad \text{otherwise} \end{cases} \tag{17}$$

where $\mu_{k_1}^n$ and $\mu_{k_2}^n$ are the average values of the k_1-th row and k_2-th column of M^n, respectively, excluding non-finite entries. The matrix is initialized as

$$M_{k_1,k_2}^0 = \mathcal{D}\left(f(\mathbf{d}_{k_1}^1), f(\mathbf{d}^2 k_2)\right) \tag{18}$$

where $f(\mathbf{d})$ is the fingerprint of the descriptor $\mathbf{d}$, and $\mathcal{D}$ is the distance function that would be used to match descriptors $\mathbf{d}_{k_1}^1$ and $\mathbf{d}_{k_2}^2$. The final distance matrix M is obtained at the end of the last iteration $\bar{n}$ by computing descriptor distances for potentially good matches only

$$M_{k_1,k_2} = \begin{cases} \mathcal{D}\left(\mathbf{d}_{k_1}^1, \mathbf{d}_{k_2}^2\right) \text{ if } M_{k_1,k_2}^{\bar{n}} \neq \infty \\ \infty \qquad \text{otherwise} \end{cases} \tag{19}$$

The total running time of the above matching procedure is the sum of two main contributions, $\mathcal{T}_{M^0}$ and $\mathcal{T}_M$, representing respectively the times needed to compute M^0 and M. For example, in the case of RootSIFT, the first contribution evaluates theoretically as $\mathcal{T}_{M^0} \propto (\alpha\, l_s) s_1 s_2$, where $\alpha = 16/128$ is the ratio between the fingerprint and descriptor lengths. Similarly, $\mathcal{T}_M \propto l_s(\beta s_1 s_2)$, where β is the proportion of surviving matches, ranging between $0.5^{2\bar{n}} = 0.0625$ (best case) and $0.5^{\bar{n}} = 0.25$ (worst case) by limiting iterations to $\bar{n} = 2$ (found experimentally to provide the best balance between computational efficiency and matching quality). Notice that unfiltered matching can also be represented in this general framework by defining a dummy fingerprint $\mathbf{f}^0 = 0$, so that $\alpha_0 \simeq 0$ and $\beta_0 = 1$. Hence, the final theoretical speedup is given by $(\alpha_0 + \beta_0)/(\alpha + \beta)$, and ranges between 267% and 533%. The theoretical speedup for all the other descriptors can be computed analogously.

Once M is computed, a greedy procedure such as that described in [44] can be used to get NNR matches from NN matches. Starting by an empty list $L = \emptyset$, all the table entries M_{k_1,k_2} are scanned for increasing distance values. Assuming that the ongoing list L contains matches of the form $(\mathbf{d}_{k_1'}^1, \mathbf{d}_{k_2'}^2)$ and that M_{k_1,k_2} is being currently checked, the list L is updated to include $(\mathbf{d}_{k_1}^1, \mathbf{d}_{k_2}^2)$ if both descriptors in the matching pair are not already used by matches already in L, i.e. $k_1 \neq k_1'$ and $k_2 \neq k_2'$. List updating stops when no finite M_{k_1,k_2} remains to be checked. Notice that L contains by construction the whole set of mutual matches as a subset. The NNR distance

$$\mathcal{D}_{\text{NNR}} = \frac{\mathcal{D}(\mathbf{d}_{k_1'}^1, \mathbf{d}_{k_2'}^2)}{\mathcal{D}(\mathbf{d}_{k_1'}^1, \mathbf{d}_{k_2''}^2)} = \frac{M_{k_1',k_2'}}{M_{k_1',k_2''}} \tag{20}$$

is eventually used to sort matches $(\mathbf{d}_{k_1'}^1, \mathbf{d}_{k_2'}^2) \in L$ in decreasing order, where $k_2'' = \underset{k_2 \neq k_2'}{\operatorname{argmin}} M_{k_1',k_2} \geq M_{k_1',k_2'}$ is the index of the 2^{nd} best match. In the place of NNR, we actually use an improved distance, referred to as symmetric NNR (sNNR):

$$\mathcal{D}_{\text{sNNR}} = \frac{2\mathcal{D}(\mathbf{d}_{k_1'}^1, \mathbf{d}_{k_2'}^2)}{\mathcal{D}(\mathbf{d}_{k_1'}^1, \mathbf{d}_{k_2''}^2) + \mathcal{D}(\mathbf{d}_{k_1''}^1, \mathbf{d}_{k_2'}^2)} \tag{21}$$

In Eq. 21, index k_1'' is defined analogously to k_2''. $\mathcal{D}_{\text{sNNR}}$ is the harmonic mean between the two values obtained from $\mathcal{D}_{\text{NNR}}$ when the 2^{nd} best is chosen from either S_1 or S_2. $\mathcal{D}_{\text{sNNR}}$ has no computational overheads with respect to $\mathcal{D}_{\text{NNR}}$, and has the remarkable advantage of taking into account the statistical context of *both* the descriptor sets S_1 and S_2. To the best of our knowledge, this is the first time that sNNR is defined and evaluated.

Table 4 mAP (%) using HCF on the planar scenes.

		$\dot{x}_i\|\dot{y}_i$	$R(y_i)$	$q_{N,3}(y_i)$
$\dot{\mathbf{f}}$	SIFT	62.72	62.80	63.22
	SIFT$_{L_1}$	63.19	62.75	62.91
$\ddot{\mathbf{f}}$	sGLOH2	73.55	74.77	75.28
$\dddot{\mathbf{f}}$	BiSIFT	60.11	–	–
	BisGLOH2	68.83	–	–

Table 5 Average running time speedup (%) for matching using HCF. Red values indicate performance loss.

			`uchar`	`float`	nibble	H
single thread	SSE 4.1	SIFT	144	160	138	–
		SIFT$_{L_1}$	56	170	99	119
		BiSIFT	–	–	–	80
		sGLOH2	267	330	258	274
		BisGLOH2	–	–	–	157
	popcount	SIFT	194	291	270	–
		SIFT$_{L_1}$	276	306	262	127
		BiSIFT	–	–	–	98
		sGLOH2	383	414	360	244
		BisGLOH2	–	–	–	194
multi thread	SSE 4.1	SIFT	142	157	136	–
		SIFT$_{L_1}$	69	165	107	117
		BiSIFT	–	–	–	81
		sGLOH2	318	367	316	316
		BisGLOH2	–	–	–	177
	popcount	SIFT	281	271	278	–
		SIFT$_{L_1}$	253	275	292	133
		BiSIFT	–	–	–	94
		sGLOH2	402	419	391	281
		BisGLOH2	–	–	–	524

H 3 bits stretched | binary (both use Hamming)

4.3 Experimental results

Results for the HCF strategy obtained with the same experimental setup of Secs. 2.3 and 3.3 are reported in terms of mAP in Table 4. Both the original descriptors and their variants were tested. Using HCF, the number of correct matches only slightly decreases for SIFT and BiSIFT (performance loss is about 1% of mAP). A more evident loss, yet negligible in practice, is found for sGLOH2 (about 3%), while BisGLOH2 exhibits the most relevant loss (about 6%).

Concerning the matching computational efficiency, Table 5 reports the measured speedup, that is the ratio between the running time of the matching procedure without HCF and with HCF, expressed in percentage. The amortized running time, i.e. the sum of the times required to compute M^0 and M divided by $s_1 \times s_2$, is considered. Values less than 100% (shown in red) indicate the unfavorable case when HCF is slowing down the computations. The purpose of HCF is actually to improve computational efficiency, and indeed it succeeds at that in all cases save for BiSIFT and SIFT$_{L_1}$ with SSE optimization on non-floating point data. This is possibly due to the relatively high computational overhead required by HCF, as compared with the most efficient matching distance implementations. The HCF speedup is more evident, attaining values up to about 300%, when either a large data type (e.g., `float`) is used, or when the L_2 distance is used without code optimization. HCF is less effective on binary descriptors, while it is especially beneficial to sGLOH2, thanks to the fact that RIFT fingerprint $\ddot{\mathbf{f}}$ is rotationally invariant.

Table 6 mAP (%) using sNNR on planar scenes with and without HCF.

		$\dot{x}_i\|\dot{y}_i$	$R(y_i)$	$q_{N,3}(y_i)$
	SIFT	64.69	65.31	65.11
	SIFT$_{L_1}$	65.04	65.22	64.80
	BiSIFT	62.90	–	–
	sGLOH2	78.21	79.42	79.24
	BisGLOH2	77.04	–	–
HCF	SIFT	63.96	64.33	64.42
	SIFT$_{L_1}$	64.39	64.31	64.05
	BiSIFT	61.02	–	–
	sGLOH2	74.61	75.87	76.25
	BisGLOH2	69.78	–	–

Table 6 reports the mAP using sNNR in the place of NNR. Better matching results are usually obtained by sNNR with respect to NNR (compare with Tables 1 and 3): An increase of about 2% in terms of mAP is generally observed, which suggests that, for the purpose of ranking matches, sNNR should always be preferred to NNR.

Table 7 reports the average matching accuracy and running times of SIFT and RootSIFT (with `uchar` and `float` data types, respectively) when different approximated matching methods are employed. Besides HCF, two ANN implementations were evaluated, i.e. the Fast Library for Approximate Nearest Neighbors (FLANN) [49] and the header-only Hierarchical Navigable Small World graphs NN search library (HNSW) [43]. For both FLANN and HNSW, several parameter setups were checked, only the most accurate (referred to as "+") and fastest ("×") of which are shown in the table (more details can be found in the additional material). Notice that HNSW is only available for matching with L_2. For ANN matching, a data structure is built from one image, and the 1^{st} and 2^{nd} nearest neighbors are retrieved for each descriptor on the other image, used as query. In the table, the total time required for building the data structure and querying are shown. Clearly, using ANN can be very efficient when several images have to be matched against a single one, since a sin-

Table 7 mAP (%) and average matching total running times (s) per image pairs of approximated matching strategies.

	approx. method	matching strategy	RootSIFT		SIFT	
			mAP	⊡	mAP	⊡
L_1	–	sNNR	60.06	0.1177	58.78	0.0339
	HCF	sNNR	59.33	0.0721	58.21	0.0617
	FLANN$_+$	NNR	53.00	0.3293	54.55	0.3287
	FLANN$_\times$	NNR	46.12	0.0186	46.64	0.0186
L_2	–	sNNR	59.88	0.1341	57.90	0.1066
	HCF	sNNR	57.43	0.0850	59.29	0.0763
	FLANN$_+$	NNR	52.59	0.3190	53.54	0.3192
	HNSW$_+$	NNR	48.97	0.0594	51.04	0.0618
	FLANN$_\times$	NNR	45.11	0.0182	45.44	0.0182
	HNSW$_\times$	NNR	23.91	0.0172	27.40	0.0174

⊡ single thread running times + best setup × fastest setup

gle data structure has to be built. On the other hand, using sNNR with ANN would imply doubling the running time, as computation of the search data structure for both the images of a given pair would be required (besides, as reported in Table 6, this would bring a gain no larger than %3). According to the results, HCF is more accurate then any ANN approach and relatively faster than the most accurate ANN equivalent methods. Using their fastest setup, both FLANN and HNSW are remarkably more efficient than HCF, yet at the expense of a considerable accuracy drop.

5 Comparative evaluation of descriptors

5.1 Related work

The purpose of tidying up the crowded panorama of local descriptors has emphasized the need of good evaluation benchmarks, exposing both the potential strengths and weaknesses of descriptors. Scene content, computational constraints, matching precision and application task all affect the choice of a descriptor. However, a fact that should be taken into account when choosing a descriptor is that even slight differences in the evaluation benchmark or in the descriptor implementation can at times lead to unclear performance results [5, 9, 52].

The most common benchmarks employ planar scenes, for which the ground truth can easily be obtained. The de facto standard Oxford benchmark [44] has been recently sided by the HPatches benchmark [5], that considers a larger dataset and variability of operational conditions. By construction, planar benchmarks are unable to take into account critical issues, such as occlusions, arising with non-planar scenes. To deal with these critical issues, benchmarks on non-planar scenes have been proposed as well over the years, including datasets with few simple 3D scenes [25] and more complex ones with either approximated [11, 47] or more refined and expensive sensor-based [21,56] ground-truths. Yet, ground-truths obtained with the approaches above are typically limited to a set of selected image regions. Quite recently, application-based evaluation benchmarks have also been introduced, attempting to indirectly infer descriptor characteristics from the expected output of an assigned visual task. These benchmarks mainly target at object retrieval tasks [22], Structure-from-Motion (SfM) [52] or visual Simultaneous Localization and Mapping (SLAM) [13] applications. Application-based benchmarks also suffer of important limitations, as they introduce a bias towards the considered application. For example, SfM-based benchmarks, that focus on keypoint localization accuracy and are generally built and optimized over SIFT, consider SIFT to be globally the best [32, 52]. On the other hand, benchmarks based on object retrieval do not rank SIFT as the best descriptor, as they usually require a higher tolerance to patch deformations [6, 9].

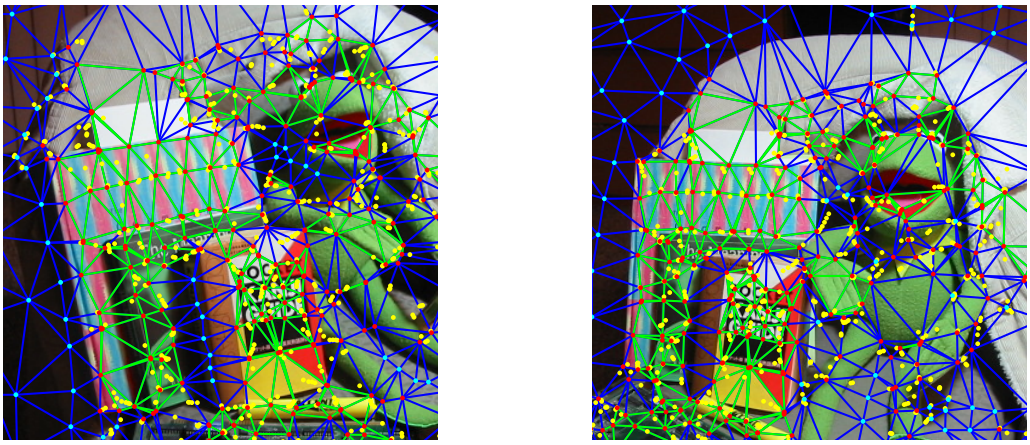
In the following, a new and general benchmark for non-planar scenes is introduced, that extends and refines the one proposed in [11] and employed in a recent descriptor evaluation [10]. This benchmark is aimed at evaluating descriptor behavior in real-world scenes, thus providing a deep insight into descriptor characteristics, while compensating for the limitations of both the planar and the application-based benchmarks.

5.2 A new benchmark for non-planar scene matching

Given a pair of images I_1 and I_2, the method requires that the user provides an initial set P of point correspondences, the generic pair being $(\mathbf{p}_1, \mathbf{p}_2)$, with $\mathbf{p}_1 \in I_1$, and $\mathbf{p}_2 \in I_2$ (see Fig. 6). A user-friendly Matlab interface to select these points is provided, together with the data and code employed in the evaluation [3].

Referring to Fig. 7, given two elliptical keypoint patches $E_1 \subset I_1$ and $E_2 \subset I_2$, the two points $\mathbf{t}_2$ and $\mathbf{t}'_2$ of tangency on E_2 from $\mathbf{p}_2$, are computed and reprojected onto I_1 as the epipolar lines $\mathbf{l}$ and $\mathbf{l}'$. The points $\mathbf{q}$ and $\mathbf{r}$ on line $\mathbf{l}$ at minimal distance with respect to the points of tangency $\mathbf{t}_1$ and $\mathbf{t}'_1$ on E_1 from $\mathbf{p}_1$, are the best candidate approximations for the mapping of $\mathbf{t}_2$ onto I_1 according to the epipolar geometry. Similarly, $\mathbf{q}'$ and $\mathbf{r}'$ on line $\mathbf{l}'$ are the best candidate approximations for the mapping of $\mathbf{t}'_2$ onto I_1. (Notice that, since two points in I_1 are actually obtained for each point of tangency in I_2, a two-fold correspondence ambiguity arises, that will be removed hereafter.) Repeating this process for all $\mathbf{p}_2$'s belonging to corresponding pairs,

[3] https://drive.google.com/open?id=1kDdToyc11QnYtH6eHr5gXYPN_Jk0tKnV



(a) (b)

Fig. 6 A zoom of the input setup in the case of non-planar scene for an example image pair I_1 (a) and I_2 (b). User-defined corresponding matches, occluded points, and keypoint centers are shown respectively in red, cyan and yellow. The blue/green wire-frames represent the Delaunay triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ when local mapping homographies are present/absent. (Best viewed in color.)

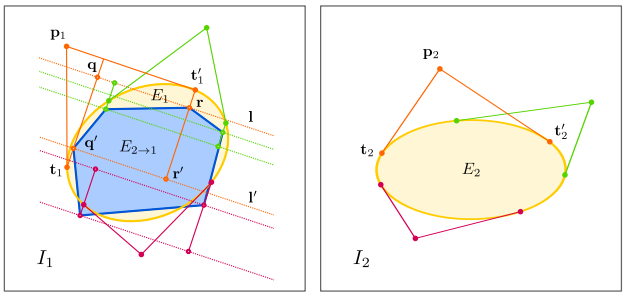


Fig. 7 Construction of the approximated reprojecting polygon $E_{2\to1}$. Distinct colors refer to different user-given correspondences $(\mathbf{p}_1, \mathbf{p}_2)$. (Best viewed in color.)

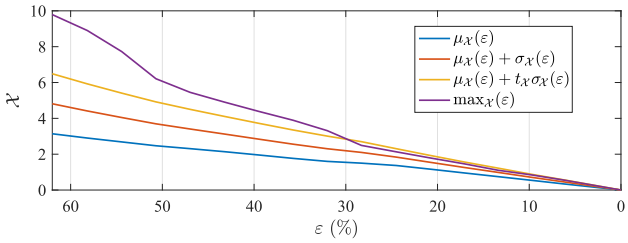


Fig. 8 $\mathcal{X}$-related statistics for decreasing values of the overlap error ε. $t_{\mathcal{X}} = 2$ in the yellow plot. (Best viewed in color.)

the best polygon $E_{2\to1}$ approximating the mapping of E_2 onto I_1 is found, where all the two-fold ambiguities are removed by considering the global minimal error solution in terms of distance between epipolar lines and tangency points. This construction is akin to the one proposed in [11] to define quadrilaterals inscribed and circumscribed to the ellipse, yet it yields a more precise ellipse reprojection, since it uses more than just two correspondences $(\mathbf{p}_1, \mathbf{p}_2)$. Analogously, a polygon $E_{1\to2} \subset I_2$ can be computed, by which the maximal approximated overlap error can be defined as

$$\varepsilon = 1 - \min\left(\frac{E_1 \cap E_{2\to1}}{E_1 \cup E_{2\to1}}, \frac{E_2 \cap E_{1\to2}}{E_2 \cup E_{1\to2}}\right) \tag{22}$$

By construction, when two patches E_1 and E_2 actually match, using ε to decide whether the two patches match would give no true negatives. Nevertheless, given two patches E_1 and E_2' that do not match, a false positive arises when E_2 (actually matching with E_1) and E_2' share, either exactly or approximately, the same tangent lines through the epipole in I_2. To alleviate this issue and increase the evaluation precision, statistics about ellipse centers were extrapolated and incorporated into the matching process. Ellipse center statistics were extrapolated with a Monte-Carlo simulation involving 4×10^7 runs. In each run, a pair of ellipses was randomly generated, then its overlap error was computed, together with the distance $\mathcal{X}$ between the two ellipse centers, normalized by the major semi-axis of one of the two ellipses. Figure 8 shows the statistics related to $\mathcal{X}$, where $\mu_{\mathcal{X}}(\varepsilon)$ and $\sigma_{\mathcal{X}}(\varepsilon)$ are respectively the mean and standard deviation of $\mathcal{X}$ for a given overlap error ε, and $t_{\mathcal{X}}$ is a user-defined constant.

The above statistics can be exploited as follows: A match between two elliptical patches E_1 and E_2 with an overlap error ε is retained only if $\varepsilon < t_\varepsilon$ (where t_ε is a threshold, experimentally set as shown later) and

$$\mathcal{X} \approx \min\left(\frac{\|\mathbf{c}_1 - \mathbf{c}_{2\to1}\|}{a_1}, \frac{\|\mathbf{c}_2 - \mathbf{c}_{1\to2}\|}{a_2}\right) \leq \mu_{\mathcal{X}}(\varepsilon) + t_{\mathcal{X}}\sigma_{\mathcal{X}}(\varepsilon) \tag{23}$$

In Eq. 23, $\mathbf{c}_1$ and $\mathbf{c}_2$ are respectively the centers of E_1 and E_2, $\mathbf{c}_{2\to1}$ is the reprojection of $\mathbf{c}_2$ onto I_1 and,

similarly, $\mathbf{c}_{1\to 2}$ is the reprojection $\mathbf{c}_1$ onto I_2; a_1, a_2 are the major semi-axes of E_1 and E_2, respectively. (Notice that, also in the case of non-planar scenes, an elliptical patch in one image is expected to be roughly elliptical after reprojection onto the other image.) In order to estimate the unknown quantities $\|\mathbf{c}_1 - \mathbf{c}_{2\to 1}\|$ and $\|\mathbf{c}_1 - \mathbf{c}_{2\to 1}\|$ appearing in Eq. 23, local homography maps or, when these do not fit the data, their nearest neighbor approximations, are employed. To this aim, the Delaunay triangulations $\mathcal{T}_1$ on I_1 and $\mathcal{T}_2$ on I_2 are constructed from the information provided by the user, i.e., the set of correspondences $P = \{(\mathbf{p}_1^k, \mathbf{p}_2^k)\}_{k=1}^K$, plus the occlusion point sets $O_1 \subset I_1$ and $O_2 \subset I_2$ (see again Fig. 6). In particular, the $\mathbf{p}_1^k$'s and O_1 are the triangle vertexes for $\mathcal{T}_1$, and similarly the $\mathbf{p}_2^k$'s and O_2 are the triangle vertexes for $\mathcal{T}_2$. Notice that in general it is not possible to transfer a triangulation from one image to another, due to the presence of parallax. The search for compatible homography maps is described hereafter. Consider the triangle $(\mathbf{p}_1^1, \mathbf{p}_1^2, \mathbf{p}_1^3)$ of $\mathcal{T}_1$ that includes the point $\mathbf{c}_1$. For each pair $\left((\mathbf{p}_1^1, \mathbf{p}_1^2, \mathbf{p}_1^3), (\mathbf{p}_1^1, \mathbf{p}_1^3, \mathbf{p}_1^4)\right)$ of adjacent triangles not containing vertexes in O_1, the homography H that maps the ordered vertex list $V_1 = \{\mathbf{p}_1^1, \mathbf{p}_1^2, \mathbf{p}_1^3, \mathbf{p}_1^4\}$ onto $V_2 = \{\mathbf{p}_2^1, \mathbf{p}_2^2, \mathbf{p}_2^3, \mathbf{p}_2^4\} \subset I_2$ is computed (see Fig. 9). The homography H is retained if

$$\left\|\mathbf{p}_2^h - \mathrm{H}^{-1}\mathbf{p}_1^h\right\| \leq t_H \tag{24}$$

where t_H is a threshold on the reprojection error experimentally set to 5 pixels, and the index h spans all correspondences for which either $\mathbf{p}_2^h \in V_2$, or (see again Fig. 9, *right*) it is the vertex of a triangle in $\mathcal{T}_2$ that intersects with triangle $(\mathbf{p}_2^1, \mathbf{p}_2^2, \mathbf{p}_2^3)$. If such intersection occurs with a triangle of $\mathcal{T}_2$ having a vertex in O_2, H is also rejected. The first of the unknown quantities in Eq. 23 can then be obtained as

$$\|\mathbf{c}_1 - \mathbf{c}_{2\to 1}\| \approx \min_{\mathrm{H}\in\mathcal{H}_{1\to 2}} \left\|\mathbf{c}_1 - \mathrm{H}^{-1}\mathbf{c}_2\right\| \tag{25}$$

provided that $\mathcal{H}_{1\to 2}$, denoting the set of the homographies (at most three) assigned to triangle $(\mathbf{p}_1^1, \mathbf{p}_1^2, \mathbf{p}_1^3)$, is not empty. The procedure can be repeated by swapping the two images, thus obtaining $\|\mathbf{c}_2 - \mathbf{c}_{1\to 2}\|$.
When $\mathcal{H}_{1\to 2} = \mathcal{H}_{2\to 1} = \emptyset$, the procedure above does not apply, and the unknowns are obtained by nearest neighbor estimation as follows. Define $P_1 \subset P$ as the set composed of vertexes $\mathbf{p}_1^f$ of triangles in $\mathcal{T}_1$ that intersect with a circle centered in $\mathbf{c}_1$ whose area is equal to the average area of the triangles in $\mathcal{T}_1$ (see Fig. 10). Considering the flow $\left(\mathbf{p}_2^f - \mathbf{p}_1^f\right)$ of any $\mathbf{p}_1^f \in P_1$, the first of the unknown quantities in Eq. 23 is estimated as

$$\|\mathbf{c}_1 - \mathbf{c}_{2\to 1}\| \approx \min_{(\mathbf{p}_1^f, \mathbf{p}_2^f)\in P_1} \left\|\mathbf{c}_1 - \left(\mathbf{c}_2 - \left(\mathbf{p}_2^f - \mathbf{p}_1^f\right)\right)\right\| \tag{26}$$

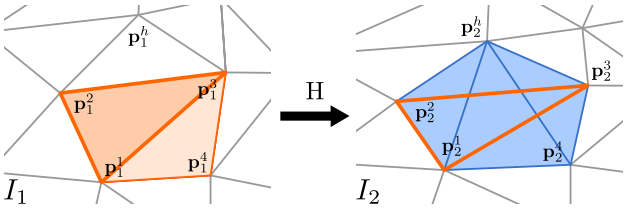


Fig. 9 The local homography H mapping quadrilateral V_1 to V_2 is assigned to triangle $(\mathbf{p}_1^1, \mathbf{p}_1^2, \mathbf{p}_1^3)$ if $V_1 \cap O_1 \neq \emptyset$, no triangle intersecting with $(\mathbf{p}_2^1, \mathbf{p}_2^2, \mathbf{p}_2^3)$ has a vertex $\mathbf{p}_2^h \in O_2$, and H^{-1} correctly reprojects $\mathbf{p}_2^h$ onto I_1. (Best viewed in color.)

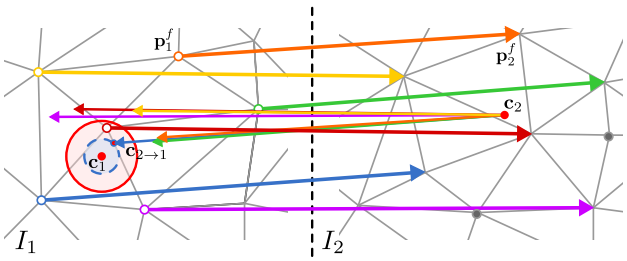


Fig. 10 In the case no local homography is found, the vertexes of the triangles intersecting with a circle centered in $\mathbf{c}_1$ (red) are considered (see text). Among the flows $\mathbf{p}_2^f - \mathbf{p}_1^f$ (colored left-to-right arrows), the one that minimizes the reprojection distance between centers $\mathbf{c}_2$ and $\mathbf{c}_1$ is selected to define $\mathbf{c}_{2\to 1}$ (in this case the blue right-to-left arrow). Occluded points are represented as gray dots. (Best viewed in color.)

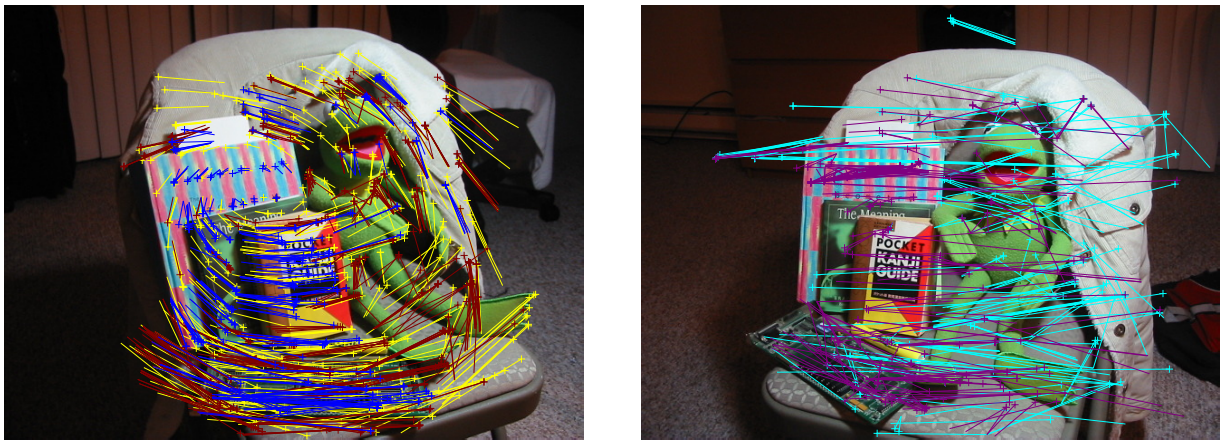
An analogous procedure is used to estimate $\|\mathbf{c}_2 - \mathbf{c}_{1\to 2}\|$.

Fig. 11 shows the estimated correct (a) and discarded (b) matches according to the proposed non-planar benchmark protocol. Clearly, the approach just discussed works with scenes with a well-defined structure and, as any other non-planar evaluation benchmark, could hardly deal with very complex and highly detailed and chaotic scenarios, such as images of dense vegetation.

5.3 Experimental results

In this section, a comprehensive comparative evaluation of local descriptors is carried out, focusing on image matching for both planar and non-planar scenes.

Results are presented for SIFT, sGLOH2 and their variants described in the previous sections (the "P" and "R" prefixes indicate respectively $q_{N,3}(y_i)$ and $R(y_i)$, RSIFT being a synonymous of RootSIFT), and for a large representative of recent state-of-the-art descriptors. These include: (a) Hand-crafted descriptors: MKD [50], LIOP [65], BRISK [35], FREAK [2], (b) Non-deep data-driven descriptors: MIOP [65], RFD [20], BRIEF [16], LATCH [36], BinBoost [60], ORB [51], BGM [61],



(a) (b)

Fig. 11 An example of non-planar scene matching output. (a) Spatial flows for (yellow) the user-given correspondences, and for keypoint matches with $\varepsilon < 0.5$ and $t_{\mathcal{X}} = 2$ when local homographies are (blue) present, (red) absent. (b) Spatial flows for discarded matches when local homographies are (purple) present or (cyan) absent. (Best viewed in color.)

LDAHash [55], (c) Deep descriptors: DeepDesc [53], HardNet [45], L2-Net and its binary variant BiL2-Net [57], Geo-Desc [41] and DOAP [27], together with its binary variant BiDOAP. Several versions of the above descriptors are tested, among those proposed by their authors. HardNetPS, i.e., the HardNet network trained with the PS dataset [46] is also included, since it is reported to perform better than the original. In the following, the 's' suffix indicates all descriptors of the same class (e.g., "SIFTs" stands for SIFT, PSIFT and RSIFT). All descriptors are matched using the sNNR matching strategy which, as noted in Sec. 4.3, performs always better than NNR (results with NNR are reported anyway in the additional material). The symbol '⇊' indicates the use of HCF. For SIFT, LIOP and GeoDesc$_Q$, results with both L_1 and L_2 distances are reported. The remaining descriptors, except for sGLOH2s (that use a permutation-based L_1), binary descriptors (that use the Hamming distance) and MKDs (that use the dot product) are optimized by construction for the L_2 distance, either by PCA or training (this also holds for GeoDesc$_Q$, yet interesting results are obtained with L_1). In order to provide a further insight into the critical steps of image matching, results using deep-based patch orientation assignment [69] in the place of the standard SIFT patch orientation approach are also given.

Results are shown in Table 8 in terms of accuracy and average matching running times, together with a short summary of the properties of each descriptor (length, data type). Running times refer to the fastest implementation of the descriptor (e.g., running times for PSIFT refer to its `uchar` implementation, see also Table 2). Notice that the time spent to get keypoint patches and to compute the descriptors is not taken into account, since in many cases descriptor code is not optimized and hardware-dependent (deep descriptors are fast on GPUs but not on CPUs, while no GPU implementation is currently available for most of the remaining descriptors). In the table, colored bands are used for a quick visualization of ranking order for the corresponding column. Different colors indicate a ranking step of 20%. For mAP, ranking refers to the best accuracy between those achieved with the two possible patch orientation assignments: SIFT ('↻') and deep-based ('↱'). For running times ranking refers to the single thread implementation. The table gives performance figures as averages over the whole datasets. More detailed results for each image pair can nevertheless be found in the additional material. Data and code used for the evaluation are freely available on-line for supporting the reproducibility of the results[3].

5.3.1 Planar scenes

For planar scene evaluation, the same datasets and setup of the previous experimental sections (see Secs. 2.3, 3.3 and 4.3) were employed. Results are provided for the Oxford and WISW datasets, that allow investigating descriptor behavior using different patch orientation assignment methods[4].

Concerning matching accuracy, significant improvements over SIFT-based orientation (about 4-8% of mAP

[4] The HPatches dataset uses square patches that cannot be rotated using patch orientation assignment without losing patch data [10]. As this would compromise the fairness of the comparison, HPatches is not considered in the present evaluation.

Table 8 Comparative evaluation of descriptors on planar and non-planar scenes. (Best viewed in color.)

		mAP (%)				matching running times (ns)					
		planar		non-planar		SSE 4.1		popcount		#	data type
		⟲	↱	⟲	↱	⊡	⊞	⊡	⊞		
L_1	SIFT	65.04	67.83	28.70	32.18	12	5	134	37	128	uchar
	SIFT$_{\Downarrow\Downarrow}$	64.39	67.12	28.23	31.77	22	7	49	15	128	uchar
	PSIFT	64.80	67.32	28.66	31.44	12	5	48	14	128	3 bits
	PSIFT$_{\Downarrow\Downarrow}$	64.05	66.59	28.25	31.68	16	7	37	11	128	3 bits
	RSIFT	65.22	68.06	30.13	33.64	38	15	163	49	128	float
	RSIFT$_{\Downarrow\Downarrow}$	64.31	67.33	29.40	32.95	23	8	54	16	128	float
	sGLOH2	–	78.21	–	39.31	418	124	1110	288	256	uchar
	sGLOH2$_{\Downarrow\Downarrow}$	–	74.61	–	36.76	157	39	290	72	256	uchar
	PsGLOH2	–	79.24	–	41.09	412	116	500	132	256	3 bits
	PsGLOH2$_{\Downarrow\Downarrow}$	–	76.25	–	38.99	160	37	205	47	256	3 bits
	RsGLOH2	–	79.42	–	42.50	670	180	1400	353	256	float
	RsGLOH2$_{\Downarrow\Downarrow}$	–	75.87	–	39.76	203	49	338	84	256	float
	LIOP	–	75.28	–	30.81	13	5	145	39	144	uchar
	GeoDesc$_Q$	79.06	80.25	40.16	43.24	12	5	134	38	128	uchar
L_2	SIFT	64.69	67.33	27.56	30.69	37	12	67	19	128	uchar
	SIFT$_{\Downarrow\Downarrow}$	63.96	66.82	27.12	30.44	26	9	35	10	128	uchar
	PSIFT	65.11	67.62	28.44	31.44	35	11	126	35	128	3 bits
	PSIFT$_{\Downarrow\Downarrow}$	64.42	66.81	28.08	31.11	25	8	47	13	128	3 bits
	RSIFT	65.36	68.13	29.34	32.61	44	15	157	43	128	float
	RSIFT$_{\Downarrow\Downarrow}$	64.33	67.49	28.61	32.07	27	9	55	16	128	float
	LIOP	–	73.44	–	28.93	39	13	70	20	144	uchar
	MIOP	–	77.59	–	32.40	45	16	160	45	128	float
	DeepDesc	55.73	59.37	29.38	33.38	44	15	156	45	128	float
	DOAP	70.37	76.38	38.29	43.76	44	15	157	44	128	float
	GeoDesc	80.25	82.61	40.92	43.89	44	15	159	45	128	float
	GeoDesc$_Q$	80.32	82.58	40.92	43.98	38	13	67	19	128	uchar
	HardNetPS	75.05	80.33	36.72	41.61	44	15	159	46	128	float
	HardNet++	71.52	77.96	38.14	43.27	45	14	156	43	128	float
	L2-Net	61.52	63.92	33.35	37.53	46	16	158	46	128	float
	L2-Net$_{CS}$	68.54	73.97	38.16	43.87	68	21	281	79	256	float
$*$	MKD$_W$	63.54	68.18	30.81	35.34	38	13	141	39	128	float
	MKD	63.71	68.20	29.97	34.35	62	20	216	64	238	float
H	BiSIFT	62.90	64.96	26.07	29.32	13	5	17	6	482	bit
	BiSIFT$_{\Downarrow\Downarrow}$	61.02	62.74	25.28	28.18	16	6	18	7	482	bit
	BisGLOH2	–	77.04	–	35.23	376	90	401	98	1152	bit
	BisGLOH2$_{\Downarrow\Downarrow}$	–	69.78	–	30.92	240	51	278	57	1152	bit
	BGM	56.06	57.21	21.88	24.04	10	3	16	5	256	bit
	BinBoost$_{64}$	33.83	30.36	11.13	11.25	8	4	9	4	64	bit
	BinBoost$_{128}$	49.57	48.36	18.16	19.34	8	3	11	4	128	bit
	BinBoost$_{256}$	56.91	57.21	21.94	23.51	9	5	16	16	256	bit
	BRIEF$_{32}$	56.11	55.89	18.25	19.15	9	4	16	5	256	bit
	BRIEF$_{64}$	57.85	58.40	19.76	20.81	13	6	25	8	512	bit
	BRISK	59.26	59.20	22.31	22.65	13	6	23	9	256	bit
	FREAK	52.87	51.87	19.25	20.39	13	5	23	8	512	bit
	LATCH$_{32}$	56.80	56.86	17.48	19.19	9	4	15	5	256	bit
	LATCH$_{64}$	60.08	60.52	19.71	21.70	13	5	24	7	512	bit
	LDAHash$_{64}$	51.19	49.17	18.51	19.16	8	3	9	3	64	bit
	LDAHash$_{128}$	58.60	58.51	22.31	23.77	7	3	11	4	128	bit
	ORB	56.60	54.82	17.32	17.39	9	4	15	6	256	bit
	RFD$_R$	68.93	70.12	29.79	28.77	11	4	18	5	293	bit
	RFD$_G$	69.43	70.67	31.22	30.37	13	5	22	8	406	bit
	BiDOAP	61.04	62.68	27.32	29.62	10	4	15	5	256	bit
	BiL2-Net	50.22	50.53	25.42	28.52	8	4	11	5	128	bit
	BiL2-Net$_{CS}$	62.76	67.53	33.82	38.35	10	5	16	6	256	bit

⟲ SIFT patch orientation ↱ deep-based patch orientation or none ⊡ single thread ⊞ multi thread
rank (best to worst) # descriptor length $*$ dot product $\Downarrow\Downarrow$ HCF

in all cases except for some binary descriptors) are obtained with deep-based patch orientation assignment. The best mAPs (■) are achieved by GeoDescs, HardNets, MIOP, sGLOH2s and BisGLOH2, followed by (■) DOAP, LIOP, L2-Net$_{CS}$, and then (■) SIFT, MKDs and BiL2-Net$_{CS}$. The proposed BiSIFT (■) performs generally better than most of the other binary descriptors (■), that rank last. The proposed quantization scheme considerably shrinks descriptor length, yet without compromising accuracy, which is even slightly increased on SIFT with L_2 and sGLOH2. It is worth noting that both SIFTs and LIOP perform slightly better with L_1 than with L_2, although the latter is the only distance considered for these descriptors in the previous literature.

5.3.2 Non-planar scenes

In order to evaluate descriptor matching according to the new benchmark protocol introduced in Sec. 5.2, 35 non-planar sequences were chosen from [1, 9, 47, 56]. In particular, 3 images were selected from 18 of these sequences, and 2 from each of the remaining 17, for a total of $(18 \times 3) + 17 = 71$ distinct image pairs (see additional material). For each pair, 450 correspondences and 380 occluded keypoints were manually selected on average, in order to estimate the ground-truth matchesas described in Sec. 5.2. The same dataset was also used for the WISW contest [10], but with a less accurate method for overlap error estimation than the one used here.

The question arises whether the precision of the proposed descriptor matching evaluation depends on the thresholds values t_ε (maximum allowed approximated overlap error) and $t_\mathcal{X}$ (patch center distance). Indeed, lower threshold values may increase the precision of correct matches, and decrease the matching tolerance. In order to show the robustness of the current evaluation in terms of ranking stability, tests were repeated considering the $3 \times 3 = 9$ possible thresholds combinations obtained from $t_\varepsilon \in \{0.4, 0.5, 0.6\}$ and $t_\mathcal{X} \in \{1, 2, 2.5\}$. Tests in terms of the Kendall τ rank correlation coefficient show that descriptor ranking keeps quite stable and does not exhibit relevant fluctuations, so that using a single threshold pair for the evaluation is sufficient to acquire a clear understanding of the relative behavior of descriptors.

Accuracy results are shown in Table 8 for $t_\varepsilon = 0.5$ and $t_\mathcal{X} = 2$. As with planar scenes, sNNR and deep patch orientation assignment are preferable to their alternatives. In terms of absolute values, mAP results generally halve for worse, clearly because non-planar scenes exhibit more inherent complexity. Descriptors ranked first (■) are GeoDescs, L2Net$_{CS}$, DOAP, HardNets and sGLOH2s. With respect to the planar case, some remarkable changes in ranking order are observed. Specifically, LIOP and MIOP fall down by two ranking classes, while DeepDesc, MKDs, DOAP and L2-Net$_{CS}$ rise up by one class, the latter two reaching first class placements. Moreover, HardNetPS gets worse, swapping with HardNet++. Relative changes for hand-crafted descriptors can be related to their degree of specialization with respect to plane-induced patch deformations. For example, LIOP and MIOP are somewhat too focused on planar transformations, hence they hardly tolerate the presence of patch deformations induced by 3D content. Changes in ranking order for deep descriptors can be related to the nature of the datasets used for training and testing them. In particular, since HardNetPS uses training data obtained by SfM, its performance loss with respect to HardNet++ is possibly due to a specific bias towards SfM applications.

5.3.3 Running times for the matching procedure

Concerning running times for the matching step, ranking order is almost the same with or without explicit SIMD code optimization (see again Table 8). The most efficient descriptors (■) are of course the binary ones, but also `uchar` SIFTs, LIOP and GeoDesc$_Q$ perform equally well, if matched with L_1. In particular, GeoDesc$_Q$, when matched with L_1 instead than with the usual L_2, quadruples the speed with negligible accuracy loss, thus ranking first both for accuracy and efficiency. Non-binary descriptors of standard design (i.e., 128 `float` vectors matched with L_2) have intermediate rankings (■), and are about three-four times slower than those above. sGLOH2s are the most computationally demanding descriptors (■).

HCF is usually effective at lowering running times by about one half, at the expense of a small accuracy drop (see e.g. the case of RootSIFT), with the only exceptions of `uchar` SIFTs matched with L_1 and BiSIFT (for an explanation, see again Sec. 4.3).

In order to achieve their fast performance, deep descriptors such as GeoDescs and HardNets heavily rely on GPU power for the descriptor computation phase. Without a GPU, their computational efficiency is very poor. For example, in our basic setup (Intel Core i7-3770K with 8 GB of RAM, no GPU), the time required by HardNet to compute 2500 descriptors is about 10 sec, to be compared with 1 sec, that is the time approximately required by SIFTs for the same task. This also implies that deep descriptors are less flexible and adaptable to different system configurations than those which are not based on deep learning.

5.4 Recapitulation and afterthoughts

From early attempts such as DeepDesc, deep descriptors have evolved and matured, so that recent descriptors based on deep learning, such as GeoDescs and HardNets, prove to provide the best accuracy for a wide class of image transformations. Nevertheless, deep descriptors are not exempt from disadvantages, including an obvious dependency on the training data and heavy requirements in terms of storage and GPU hardware, by which to achieve reasonable running times. In the authors' opinion, the matching accuracy improvements offered by deep descriptors do not fully compensate for their lack of flexibility in practical scenarios, and universality.

SIFT is confirmed to be the reference general-purpose descriptor, being still able to compete, despite its age, with recent non-binary hand-crafted descriptors (MKDs, MIOP and LIOP) [52]. As shown in this paper, SIFT can be made even more competitive and appealing by packing it into PSIFT according to the introduced quantization scheme. Although being only 48 bytes long, PSIFT does not exhibit performance losses with respect to SIFT. The proposed BiSIFT also yields very good results, both in accuracy and efficiency, as compared with most state-of-the-art binary descriptors. RootSIFT, currently the most accurate yet slowest among the SIFT variants, can be made twice as fast by using the proposed HCF approach. HCF is also a good way to improve efficiency for all SIFTs using L_2 as matching distance, and sGLOH2s.

As evident from our experimental results, the best trade-off between accuracy and efficiency is obtained with `uchar` descriptors matched with the L_1 distance. Exemplary in this respect is GeoDesc$_Q$ which, if matched with L_1 instead than with the usual L_2, quadruples the speed with negligible accuracy loss. A similar behavior has to be expected for all descriptors. Indeed, on the one hand, L_1 and L_2 give comparable values for all high-dimensional descriptor vectors, which explains why the accuracy loss is minimal. On the other hand, the maximal computational efficiency is obtained with the SAD instruction, which implements L_1 and requires that the descriptor be represented as a `uchar`. The above observation could be taken into account when designing novel non-binary descriptors and making them as fast as binary descriptors and as accurate as their `float`, L_2-matched counterparts.

Exploiting data context at running time instead of injecting data knowledge into the descriptor at training time can offer new solutions for matching, which were not entirely explored so far (geometric constraints [12, 24, 38] also share the same objective, but are less general). This is the idea behind the design of the matching strategy sNNR and the descriptor class derived from sGLOH2, that is essentially a rotating SIFT. sGLOH2s are very stable and robust, and currently are the only handcrafted descriptors comparable with deep descriptors in terms of matching accuracy.

According to our evaluation, patch orientation always plays a key role in descriptor design. In particular, the more correct matches are needed, the finer the orientation registration has to be, save for those descriptors (i.e., sGLOH2s, LIOP and MIOP) embedding a mechanism to handle rotational invariance.

6 Conclusions and future work

This paper addressed the design and evaluation of computational strategies aimed at enhancing image matching with SIFT. We have undertaken a critical review of the main aspects of descriptor design that affect matching performance, and devised new ways to do old things better: From quantization and binarization, through contextual matching, to experimental benchmarking. Most of the enhancements are actually SIFT-independent, and can be employed with any hand-crafted or data-driven descriptors, also including deep descriptors. Moreover, we have produced a thorough experimental analysis, including a very comprehensive comparison of baseline and state-of-the-art descriptors. From both the theoretical and experimental discoveries made, the following conclusions can be drawn, which we hope will prove useful to practitioners in the field, who are invited to download our code and data[3].

- **Quantization can improve spatial and temporal descriptor computational efficiency without compromising matching accuracy** *(Sec. 2)*. A novel quantization scheme for SIFT was introduced. The quantized descriptor, called PSIFT, at least halves the number of bits required to store the descriptor, yet without any significant loss in terms of both accuracy and efficiency. The 48-bytes PSIFT can also be recoded into a 128-bytes binary descriptor with no loss of accuracy, with improved efficiency when no optimized code is available.
- **Comparing and sorting are strictly related in binary descriptor design** *(Sec. 3)*. An alternative binarization scheme for SIFT was investigated, resulting in a light descriptor (BiSIFT), competitive with other state-of-the-art binary descriptors. Experimental evidence with BiSIFT provided an insight into the benefits of comparing against sorting in binary descriptor design.

- **A hierarchical coarse-to-fine descriptor representation can improve matching efficiency in the case of heavy distance computation** *(Sec. 4)*. A hierarchical cascade filtering (HCF) approach was proposed that, relying on the extraction of small descriptor fingerprints and exploiting the multi-level structure of SIFT-like descriptors, can further speedup the descriptor matching process, especially with `float` descriptors matched with the L_2 distance.
- **One-side distances may be not optimal when matching a pair of images** *(Sec. 4)*. A symmetrical variant of NNR (sNNR) was defined, and shown experimentally to increase the matching accuracy.
- **Designing benchmarks for non-planar scenes is not an easy task** *(Sec. 5)*. A novel evaluation benchmark extending the concept of patch overlap to the non-planar case was developed, allowing us to compare descriptor behavior on both planar and non-planar scenes.
- **Descriptor ranking may change even significantly from planar to non-planar scenes** *(Sec. 5)*. In particular, training data for deep descriptors should be chosen carefully, in order to avoid any bias towards a specific scene geometry.
- **Deep descriptors achieve the best matching accuracy, SIFT is still well balanced overall** *(Sec. 5)*. According to the evaluation, deep descriptors are currently those that perform best in terms of accuracy, and can also be computationally efficient on GPUs. SIFT-like descriptors, especially if properly quantized and matched, remain competitive still today in terms of balance between accuracy, storage, efficiency and hardware-software flexibility.
- **Patch orientation is critical for matching** *(Sec. 5)*. This is evidenced from the evaluation by comparing the canonical SIFT orientation estimation against the deep-based orientation estimation.
- **Local and global descriptor contexts both matter in image matching** *(Secs. 4 and 5)*. This is corroborated by the overall experimental results with HCF, sNNR and patch orientation.
- **The `uchar` descriptors matched with the L_1 distance yield at the same time the computational efficiency of binary descriptors and the high accuracy of `float` descriptors matched with the L_2 distance** *(Secs. 2 and 5)*. This should be kept in mind when designing future descriptors.

Future work will address the problem of deep descriptor quantization, as it plays a key role in encoding any descriptor into `uchar` format, but also in view of the recent `bfloat16` half precision floating-point format[5]. Data context exploitation appears to be another promising direction for future research in image matching. Descriptor matching on GPUs is becoming common practice. However, current implementations are limited to `float` descriptors matched with the L_2 distance. Therefore, another topic of future investigation will be the design of algorithms for GPUs leveraging quantization and the L_1 distance for maximizing bandwidth and speeding-up computations.

Acknowledgements The Titan Xp used for this research was generously donated by the NVIDIA Corporation.

This work is based on research partially sponsored by the Air Force Research Laboratory and the Defense Advanced Research Projects Agency under agreement number FA8750-16-2-0188. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

F. Bellavia is currently funded by the Italian Ministry of Education and Research (MIUR) under the program PON Ricerca e Innovazione 2014-2020, cofunded by the European Social Fund (ESF), CUP B74I18000220006, id. proposta AIM 1875400, linea di attività 2, Area Cultural Heritage.

References

1. Aanæs, H., Dahl, A.L., Pedersen, K.S.: Interesting interest points. International Journal of Computer Vision **97**(1), 18–35 (2012)
2. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: Fast retina keypoint. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 510–517 (2012)
3. Arandjelović, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2911–2918 (2012)
4. Baber, J., Dailey, M., Satoh, S., Afzulpurkar, N., Bakhtyar, M.: BIG-OH: BInarization of gradient orientation histograms. Image and Vision Computing **32**(11), 940–953 (2014)
5. Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K.: HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3852–3861 (2017)
6. Balntas, V., Riba, E., Ponsa, D., Mikolajczyk, K.: Learning local feature descriptors with triplets and shallow convolutional neural networks. In: Proceedings of the British Machine Vision Conference (BMVC), pp. 119.1–119.11 (2016)
7. Balntas, V., Tang, L., Mikolajczyk, K.: Bold - binary online learned descriptor for efficient image matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2367–2375 (2015)
8. Bay, H., Ess, A., Tuytelaars, T., L.Van Gool: Speeded-up robust features (SURF). Computer Vision and Image Understanding **110**(3), 346–359 (2008)
9. Bellavia, F., Colombo, C.: Rethinking the sGLOH descriptor. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(4), 931–944 (2018)

[5] `https://cloud.google.com/tpu/docs/bfloat16`

10. Bellavia, F., Colombo, C.: "Which is Which?" Evaluation of local descriptors for image matching in real-world scenarios. In: International Conference on Computer Analysis of Images and Patterns (CAIP), pp. 299–310 (2019). URL http://cvg.dsi.unifi.it/wisw.caip2019
11. Bellavia, F., Valenti, C., Lupascu, C.A., Tegolo, D.: Approximated overlap error for the evaluation of feature descriptors on 3D scenes. In: Proceedings of the International Conference on Image Analysis and Processing (ICIAP), pp. 270–279 (2013)
12. Bian, J., Lin, W.Y., Matsushita, Y., Yeung, S.K., Nguyen, T.D., Cheng, M.M.: GMS: grid-based motion statistics for fast, ultra-robust feature correspondence. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2828–2837 (2017)
13. Bian, J., Zhang, L., Liu, Y., Lin, W.Y., Cheng, M.M., Reid, I.D.: Image matching: An application-oriented benchmark. In: arXiv (2018)
14. Brown, M., Lowe, D.G.: Automatic panoramic image stitching using invariant features. International Journal of Computer Vision **74**(1), 59–73 (2007)
15. Brown, M.A., Hua, G., Winder, S.A.J.: Discriminative learning of local image descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(1), 43–57 (2011)
16. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: Binary robust independent elementary features. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 778–792 (2010)
17. Chandrasekhar, V., Takacs, G., Chen, D.M., Tsai, S.S., Reznik, Y., Grzeszczuk, R., Girod, B.: Compressed histogram of gradients: A low-bitrate descriptor. International Journal of Computer Vision **96**(3), 384–399 (2012)
18. Cui, Y., Hasler, N., Thormählen, T., Seidel, H.: Scale invariant feature transform with irregular orientation histogram binning. In: Proceedings of the International Conference on Image Analysis and Recognition (ICIAR), pp. 258–267 (2009)
19. Dong, J., Soatto, S.: Domain-size pooling in local descriptors: DSP-SIFT. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
20. Fan, B., Kong, Q., Trzcinski, T., Wang, Z., Pan, C., Fua, P.: Receptive fields selection for binary feature description. IEEE Transactions on Image Processing **26**(6), 2583–2595 (2014)
21. Fan, B., Kong, Q., Wang, X., Wang, Z., Xiang, S., Pan, C., Fua, P.: A performance evaluation of local features for image based 3d reconstruction. In: arXiv (2018)
22. Fan, B., Wu, F., Hu, Z.: Rotationally invariant descriptors using intensity order pooling. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(10), 2031–2045 (2012)
23. Fanfani, M., Bellavia, F., Colombo, C.: Accurate keyframe selection and keypoint tracking for robust visual odometry. Machine Vision and Applications **27**(6), 833–844 (2016)
24. Frahm, J., Matas, J., Pollefeys, M., Chum, O., Raguram, R.: Usac: A universal framework for random sample consensus. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 2022–2038 (2013)
25. Fraundorfer, F., Bischof, H.: A novel performance evaluation method of local detectors on non-planar scenes. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 33–33 (2005)
26. Hassner, T., Mayzels, V., Zelnik-Manor, L.: On SIFTs and their scales. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1522–1528 (2012)
27. He, K., Lu, Y., Sclaroff, S.: Local descriptors optimized for average precision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
28. Heikkila, M., Pietikainen, M., Schmid, C.: Description of interest regions with local binary patterns. Pattern Recognition **42**(3), 425–436 (2009)
29. Hua, G., Brown, M., Winder, S.: Discriminant embedding for local image descriptors. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), vol. 0, pp. 1–8 (2007)
30. Jin, Z., Zhang, D., Hu, Y., Lin, S., Cai, D., He, X.: Fast and accurate hashing via iterative nearest neighbors expansion. IEEE Transactions on Cybernetics **44**(11), 2167–2177 (2014)
31. Ke, Y., Sukthankar, R.: PCA-SIFT: a more distinctive representation for local image descriptors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 506–513 (2004)
32. Khan, N., Mccane, B., Mills, S.: Better than SIFT? Machine Vision and Applications **26**(6), 819–836 (2015)
33. Lazebnik, S., Schmid, C., Ponce, J.: A sparse texture representation using local affine regions. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(8), 1265–1278 (2005)
34. Lenc, K., Matas, J., Mishkin, D.: A few things one should know about feature extraction, description and matching. In: Proceedings of the Computer Vision Winter Workshop (CVWW), pp. 67–74 (2014)
35. Leutenegger, S., Chli, M., Siegwart, R.: BRISK: Binary robust invariant scalable keypoints. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2011)
36. Levi, G., Hassner, T.: LATCH: Learned arrangements of three patch codes. In: IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1–9 (2016)
37. Lin, K., Lu, J., Chen, C.S., Zhou, J., Sun, M.T.: Unsupervised deep learning of compact binary descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
38. Lin, W.Y., Wang, F., Cheng, M.M., Yeung, S.K., Torr, P.H.S., Do, M.N., Lu, J.: Code: Coherence based decision boundaries for feature correspondence. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(1), 34–47 (2018)
39. Ling, H., Okada, K.: Diffusion distance for histogram comparison. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 246–253 (2006)
40. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
41. Luo, Z., Shen, T., Zhou, L., Zhu, S., Zhang, R., Yao, Y., Fang, T., Quan, L.: Geodesc: Learning local descriptors by integrating geometry constraints. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)
42. Ma, J., Zhao, J., Jiang, J., Zhou, H., Guo, X.: Locality preserving matching. International Journal of Computer Vision **127**(5), 512–531 (2019)
43. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. arXiv (2016). URL https://github.com/nmslib/hnswlib/

44. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(10), 1615–1630 (2005)
45. Mishchuk, A., Mishkin, D., Radenovic, F., Matas, J.: Working hard to know your neighbor's margins: Local descriptor learning loss. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS), pp. 4829–4840 (2017)
46. Mitra, R., Doiphode, N., Gautam, U., Narayan, S., Ahmed, S., Chandran, S., Jain, A.: A large dataset for improving patch matching. In: arXiv (2018)
47. Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3d objects. International Journal of Computer Vision **73**(3), 263–284 (2007)
48. Morel, J., Yu, G.: ASIFT: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences **2**(2), 438–469 (2009)
49. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence **36** (2014). URL `https://www.cs.ubc.ca/research/flann/`
50. Mukundan, A., Tolias, G., Chum, O.: Multiple-kernel local-patch descriptor. In: British Machine Vision Conference (BMVC) (2017)
51. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571 (2011)
52. Schönberger, J.L., Hardmeier, H., Sattler, T., Pollefeys, M.: Comparative evaluation of hand-crafted and learned local features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
53. Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015)
54. Snavely, N., Seitz, S., Szeliski, R.: Modeling the world from internet photo collections. International Journal of Computer Vision **80**(2), 189–210 (2008)
55. Strecha, C., Bronstein, A., Bronstein, M., Fua, P.: LDA-Hash: Improved matching with smaller descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(1), 66–78 (2012)
56. Strecha, C., von Hansen, W., Gool, L.J.V., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
57. Tian, Y., Fan, B., Wu, F.: L2-Net: deep learning of discriminative patch descriptor in euclidean space. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6128–6136 (2017)
58. Tola, E., Lepetit, V., Fua, P.: Daisy: an efficient dense descriptor applied to wide baseline stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(5), 815–830 (2010)
59. Treen, G., Whitehead, A.: Efficient SIFT matching from keypoint descriptor properties. In: Proceedings of the Workshop on Applications of Computer Vision (WACV), pp. 1–7 (2009)
60. Trzcinski, T., Christoudias, M., Fua, P., Lepetit, V.: Boosting binary keypoint descriptors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2874–2881 (2013)
61. Trzcinski, T., Christoudias, M., Lepetit, V., Fua, P.: Learning image descriptors with the boosting-trick. In: Advances in neural information processing systems, pp. 269–277 (2012)
62. Tsai, C., Tsao, A., Wang, C.: Real-time feature descriptor matching via a multi-resolution exhaustive search method. Journal of Software **8**(9), 2197–2201 (2013)
63. Tuytelaars, T., Schmid, C.: Vector quantizing feature space with a regular lattice. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2007)
64. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/ (2008)
65. Wang, Z., Fan, B., Wang, G., Wu, F.: Exploring local and overall ordinal information for robust feature description. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(11), 2198–2211 (2016)
66. Wei, X., Zhang, Y., Gong, Y., Zheng, N.: Kernelized subspace pooling for deep local descriptors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
67. Winder, S., Hua, G., Brown, M.: Picking the best DAISY. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 178–185 (2009)
68. Yang, T., Lin, Y., Chuang, Y.: Accumulated stability voting: A robust descriptor from descriptors of multiple scales. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 327–335 (2016)
69. Yi, K., Verdie, Y., Fua, P., Lepetit, V.: Learning to assign orientations to feature points. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2016)
70. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. International Journal of Computer Vision **73**(2), 213–238 (2007)
71. Zhang, W., Kosecka, J.: Generalized ransac framework for relaxed correspondence problems. In: Third International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 854–860 (2006)
72. Ziegler, A., Christiansen, E., Kriegman, D., Belongie, S.: Locally uniform comparison image descriptor. In: Advances in Neural Information Processing Systems (NIPS), pp. 1–9 (2012)