

# Explicit Spatial Encoding for Deep Local Descriptors

Arun Mukundan      Giorgos Tolias      Ondřej Chum  
Visual Recognition Group, FEE, CTU in Prague

## Abstract

*We propose a kernelized deep local-patch descriptor based on efficient match kernels of neural network activations. Response of each receptive field is encoded together with its spatial location using explicit feature maps. Two location parametrizations, Cartesian and polar, are used to provide robustness to a different types of canonical patch misalignment. Additionally, we analyze how the conventional architecture, i.e. a fully connected layer attached after the convolutional part, encodes responses in a spatially variant way. In contrary, explicit spatial encoding is used in our descriptor, whose potential applications are not limited to local-patches. We evaluate the descriptor on standard benchmarks. Both versions, encoding  $32 \times 32$  or  $64 \times 64$  patches, consistently outperform all other methods on all benchmarks. The number of parameters of the model is independent of the input patch resolution.*

## 1. Introduction

Local feature extraction and representation is still an essential part of a number computer vision applications across many different problems. A common and well performing procedure is a sequence of three steps: local feature detection [16, 27, 29, 6, 25], local patch rectification into a canonical form, and finally a descriptor construction from the canonical patch [28, 6, 25, 14]. The desired property of the local patch descriptors is that Euclidean distance or a dot product between two descriptors indicates whether they are matching, i.e. the local features are coming approximately from the same surface of a 3D scene. The descriptor methods have shifted from hand-crafted to currently the most successful convolutional neural network (CNN) based approaches [58, 46, 5, 30, 49, 33].

Fully convolutional neural networks takes an image or a patch as input and produces a tensor, where a vector at each spatial location can be seen as a detector response over its receptive field. In the case of variable-sized, or non-aligned input, such as images, the response tensor is transformed into a descriptor typically by some form of global pooling [40, 19, 39], which discards geometric information. The

global pooling is analogous to bags-of-features [13, 48] or descriptor aggregation [37, 18]. In the case of aligned input of fixed size, such as rectified image patches, the tensor is vectorized and further processed. Vectorization has similar interpretation to vectorizing spatial bins in SIFT [25]. Commonly, the vectorized tensor is processed by a single fully-connected (FC) layer [30, 49], that can be either interpreted as learned affine (linear and bias) transformation of the space, e.g. whitening and dimensionality reduction, or as spatially dependent embedding with efficient match kernels (EMK) [9, 11] (see Section 3.2). The key contribution of this work is a CNN module that explicitly models the spatial information of a rectified patch. Its applicability is not limited to local descriptors.

Two rectified patches coming from matching local features are far from being identical in general. The difference has two sources, namely appearance change in imaging process and geometric misalignment. The former comes from different light conditions, non-planarity of the surface, imaging artifacts, etc. The latter is caused by the detected feature covering slightly different area of the 3D surface, or incorrect rectification of the patch. These are consequences of either the appearance changes or of insufficient geometric invariance of the detector, i.e. affine invariant detector acting on projectively transformed surface.

Prior work on hand-crafted feature descriptors has shown that it is beneficial to explicitly address the geometric misalignment. Some of the approaches handling this are soft assignment of gradients to bins in SIFT and continuous spatial encoding by kernel methods in different [11] or multiple [32] coordinate systems.

CNNs are powerful in modeling the appearance variance, while weak in modeling the geometric displacement (at least with a single FC layer). Recent methods propose different ways of incorporating spatial information in a CNN [34, 24], but their application field is different than local descriptors. In this work, we propose to model the geometric misalignment by efficient match kernels that explicitly encode the spatial positions of the responses. To encode the spatial information, kernel-based explicit feature maps are used in a similar fashion to hand-crafted features [11, 32]. This can be seen as a transition from soft

binning, *i.e.* overlapping receptive fields, to continuous efficient match kernels. In contrast to models with an FC layer, with efficient match kernels the number of model parameters does not grow with increased resolution of the input patch, *i.e.* the models for  $32 \times 32$  patch input has the same number of parameters as the model for  $64 \times 64$ . The applications of the proposed descriptor go beyond that of local patches, *e.g.* tasks where encoding spatial position is essential [24, 34].

The rest of the paper is organized as follows. The related work is discussed in Section 2. Conventional deep local descriptors and the proposed ones is discussed in Section 3. Implementation details are detailed in Section 4. Finally, we present and discuss our experiments on standard benchmarks in Section 5.

## 2. Related work

In this section, we review prior work related to hand-crafted and learned descriptors of local features.

### 2.1. Hand-crafted descriptors

There are numerous approaches to hand-craft local descriptors. The variants are based on different types of processing of the input patch, such as filter-bank responses [6, 10, 22, 36, 43], pixel gradients [25, 28, 50, 1], pixel intensities [45, 12, 23, 41] and ordering or ranking of pixel intensities [35, 17]. The most prominent direction has been that of gradient histograms, an approach followed also by the most popular hand-crafted local descriptor, namely SIFT [25]. Several improvements and extensions exist in the literature [20, 57, 21, 44, 2, 14]. The RootSIFT [2] variant efficiently estimates Hellinger distance and became a standard choice in approaches and tasks.

Kernel descriptors are derived from the concept of efficient match kernels [9] and form a flexible way to design descriptors with the desired invariant properties. Kernel descriptors have been proposed not only for local patches [11] but also as a global image descriptor [8, 7]. The kernel descriptor of Bursuc *et al.* [11] was shown to outperform learned descriptors at that time.

### 2.2. Learning descriptors

Structure-from-Motion and datasets such as PhotoTourism [55] gave rise to learned local descriptors. The learned part varies from their pooling regions [55, 47] and filter banks [55] to transformations for dimensionality reduction [47] and embeddings [38].

Learning is also applied to kernelized descriptors as in the supervised framework by Wang *et al.* [54]. The local descriptors in their case are not used separately but directly aggregated into a global image representation, while supervision comes at image level. Kernel local descriptors are

combined with supervised learning in the form of discriminative projections in the work of Mukundan *et al.* [32]. Our work is inspired by theirs; we use the same kernel-based position encoding, but on top of convolutional activations instead of pixel attributes.

### 2.3. Deep learning of descriptors

The interest in local descriptor learning is lately dominated by deep learning [46, 58, 15, 56, 5, 3]. All examples in the literature use architectures that consist of a sequence of common CNN layers, similar to the ones of generic computer vision tasks, such as object recognition, but less deep and with fewer parameters in total. They typically require a large amount of training data in the form of local patch pairs or triplets. Some of the contributions are about mining hard training samples [46, 30, 26], different loss functions [5], different architectures [49] or training jointly with the local feature detector [56].

Two of the most recent and successful deep local descriptors are L2-Net [49] and HardNet [30]. L2-Net applies the loss function to intermediate feature maps too and the loss function integrates multiple attributes. HardNet extends L2-Net by sampling the hardest within batch samples and currently constitutes the state-of-the-art descriptor. Their common characteristic, which is shared among all ancestors, is that they are using common CNN layers in their architecture. As a consequence, spatial information of convolutional feature maps is not explicitly encoded, but only processed with a standard FC layer.

## 3. Method

We initially present the current typical architecture for deep local descriptors in the literature. Then, we provide a different perspective that formulates such descriptors as match kernels. It allows us to point out how the encoding of convolutional feature maps is performed in a translation variant way, but without explicitly encoding the spatial information. Finally, we present our novel deep local descriptor which is derived through the same match kernel framework and improves exactly this drawback. We get inspired by hand-crafted kernel descriptors to incorporate explicit position encoding into deep networks for local descriptors. An overview of the proposed descriptor is shown in Figure 1,

### 3.1. Deep local descriptors

Conventional architectures for deep local descriptors consist of a sequence of convolutional layers, producing translation invariant feature maps, and a final FC layer. We denote the descriptor extraction process by function  $\psi : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^D$ , where  $N$  is the size of the input patch and  $D$  the dimensionality of the final descriptor. Descriptor

for patch  $a \in \mathbb{R}^{N \times N}$  is given by  $\psi(a) \in \mathbb{R}^D$  or equivalently  $\psi_a$  to simplify the notation.

We denote the convolutional part of the network, *i.e.* a *Fully Convolutional Network* (FCN), by function  $\phi : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{n \times n \times d}$ . Size  $n$  of the resulting feature map is related to input size  $N$  and the architecture of the network. Feature map  $\phi(a)$ , equivalently denoted by  $\phi_a$ , is a 3D tensor of activations, which we also view as a 2D grid of  $d$ -dimensional vectors. We call these vectors *convolutional descriptors* and use  $\phi_a^p$  to denote the vector with coordinates  $p = (i, j)$  on the  $n \times n$  grid, *i.e.*  $p \in [n]^2$ <sup>1</sup>. Each convolutional descriptor corresponds to a region of the input patch  $a$  that is equal to the receptive field size of the feature map.

The standard practice is to vectorize 3D tensor  $\phi_a$  and feed it to an FC layer with parameters that consist of matrix  $W \in \mathbb{R}^{D \times (n \times n \times d)}$  and bias  $\mathbf{w} \in \mathbb{R}^D$ . The final descriptor is constructed as

$$\psi_a = W \text{vec}(\phi_a) + \mathbf{w}, \quad (1)$$

where  $\text{vec}$  denotes tensor vectorization. A local descriptor is typically  $\ell_2$ -normalized, which is equivalently achieved by introducing a normalization factor  $\gamma_a = 1/\sqrt{\psi_a^\top \psi_a}$  producing descriptor  $\hat{\psi}_a = \gamma_a \psi_a$ .

Similarity (or distance) between patches  $a$  and  $b$  is estimated with inner product (or Euclidean distance)  $\hat{\psi}_a^\top \hat{\psi}_b$ . The  $\ell_2$ -normalized descriptor is always used to compare patches, but we often use  $\psi_a$  (and not  $\hat{\psi}_a$ ) simply to specify which descriptor variant is used. Several deep local descriptors in the recent literature, namely L2Net [49], HardNet [30], and GeoDesc [26] follow such an architecture and can be formulated in the same way.

### 3.2. A match-kernel perspective

We provide an alternative, but equivalent, construction of deep local descriptors. We consider matrix  $W$  as a concatenation of  $n^2$  matrices, *i.e.*

$$W = \begin{pmatrix} W_{(1,1)}^\top \\ \vdots \\ W_{(i,j)}^\top \\ \vdots \\ W_{(n,n)}^\top \end{pmatrix}^\top, \quad (2)$$

where  $W_p \in \mathbb{R}^{D \times d}$ . Descriptor in (1) can be now written as

$$\psi_a = \sum_{p \in [n]^2} W_p \phi_a^p + \mathbf{w}', \quad (3)$$

<sup>1</sup> $[i] = \{1 \dots i\}$  and  $[i]^2 = [i] \times [i]$

where  $\mathbf{w}' = \mathbf{w}/n^2$ . Moreover, patch similarity becomes

$$\begin{aligned} \hat{\psi}_a^\top \hat{\psi}_b &\propto \sum_{p,q \in [n]^2} (W_p \phi_a^p + \mathbf{w}')^\top (W_q \phi_b^q + \mathbf{w}') \\ &= \sum_{p,q \in [n]^2} g_{fc}(\phi_a^p, p)^\top g_{fc}(\phi_b^q, q), \end{aligned} \quad (4)$$

where  $g_{fc} : \mathbb{R}^d \times [n]^2 \rightarrow \mathbb{R}^D$  is a function that encodes a convolutional descriptor in a translation variant way, depending on its position in the  $n \times n$  grid. The match kernel formulation in (4) interprets deep local descriptor similarity as similarity accumulation for all pairs of positions on the  $n \times n$  grid. It reveals that matching between convolutional descriptors in  $\phi_a$  and  $\phi_b$  is performed in a translation variant way. The *encoding function*  $g$  in the case of conventional deep local descriptors is

$$g_{fc}(\mathbf{v}, p) = W_p \mathbf{v} + \mathbf{w}', \quad (5)$$

where matrix  $W_p$  and  $\mathbf{w}'$  come from the parameters of the FC layer. In this work, we propose a new encoding function  $g$ , not restricted to standard CNN architecture (layers), that explicitly encodes position  $p$  on the 2D grid.

### 3.3. Position encoding

Explicit feature maps [53] are used to encode the position. Let  $f : \mathbb{R} \rightarrow \mathbb{R}^{2s+1}$  be a feature map, where  $s$  is a design choice defining the dimensionality of the embedding. Such a feature map defines a shift invariant kernel  $K : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$  with kernel signature  $k$ , so that  $K(\alpha, \beta) = k(\alpha - \beta)$

$$f(\alpha)^\top f(\beta) = K(\alpha, \beta) = k(\alpha - \beta). \quad (6)$$

The kernel  $K$  (or the feature map  $f$ ) is constructed to approximate the Von Mises kernel [51].

We propose encoding function  $g_{xy} : \mathbb{R}^d \times [n]^2 \rightarrow \mathbb{R}^{D(2s+1)^2}$  given by

$$g_{xy}(\phi_a^p, p) = \phi_a^p \otimes f(x_p) \otimes f(y_p), \quad (7)$$

where  $\otimes$  is the Kronecker product and  $x_p$  and  $y_p$  provide the coordinates of position  $p$  in a Cartesian coordinate system<sup>2</sup>. It is a joint encoding of the convolutional descriptor and the explicit representation of its position. It is inspired by the work of Mukundan *et al.* [32] who propose a hand-crafted local descriptor that encodes pixel gradients with their positions in the patch. Similarity of two such encodings is given by

$$g_{xy}(\phi_a^p, p)^\top g_{xy}(\phi_b^q, q) = \phi_a^p^\top \phi_b^q \cdot k(x_p - x_q) \cdot k(y_p - y_q). \quad (8)$$

<sup>2</sup>For  $p = (i, j)$ ,  $x_p = i$  and  $y_p = j$ .

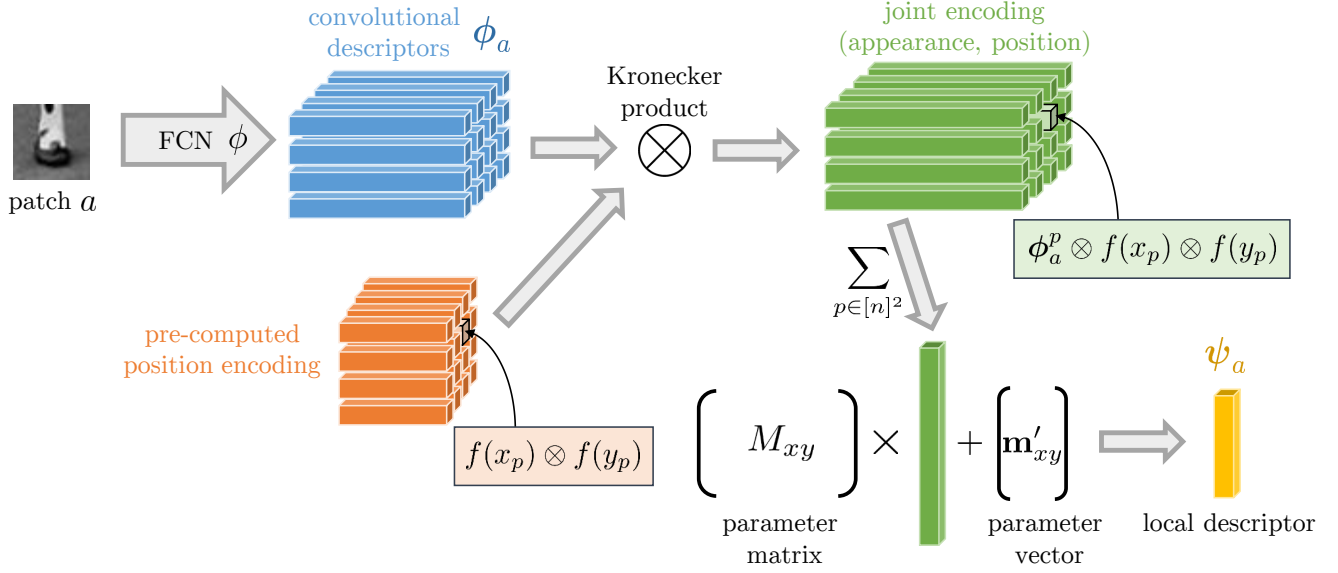


Figure 1. Overview of extraction process for the proposed descriptor. We present the case of  $\psi^{xy}$  (10), while other variants are performed in a similar way.  $\mathbf{m}'_{xy} = n^2 \mathbf{m}_{xy}$ .

It is equivalent to the product of descriptor similarity and similarity of positions on the Cartesian grid.

Following the paradigm of descriptor whitening of hand-crafted descriptors [32, 4], we propose the final local descriptor

$$\psi_a^{xy} = \sum_{p \in [n]^2} w_p M_{xy} g_{xy}(\phi_a^p, p) + \mathbf{m}_{xy} \quad (9)$$

$$= M_{xy} \left( \sum_{p \in [n]^2} w_p g_{xy}(\phi_a^p, p) \right) + n^2 \mathbf{m}_{xy}, \quad (10)$$

where  $M_{xy} \in \mathbb{R}^{D \times d(2s+1)^2}$  and  $\mathbf{m}_{xy} \in \mathbb{R}^D$  are parameters to be learned during training, while  $w_p = \exp(-\rho_p^2)$  is a weight giving importance according to the distance  $\rho_p$  from the center of the patch. Note that in contrast to (3) the same matrix, *i.e.*  $M_{xy}$ , is used for all convolutional descriptors. As a result the number of required parameters is reduced and multiplication by  $M_{xy}$  can be efficiently performed after the summation (10). In analogy to the encoding of position in a Cartesian coordinate system, we additionally propose the encoding w.r.t. a polar coordinate system<sup>3</sup> by

$$g_{\rho\theta}(\phi_a^p, p) = \phi_a^p \otimes f(\rho_p) \otimes f(\theta_p), \quad (11)$$

<sup>3</sup>For  $p = (i, j)$ ,  $\rho_p = \sqrt{(i-c)^2 + (j-c)^2}$  and  $\theta_p = \tan^{-1} \frac{j-c}{i-c}$ , where  $c = (n+1)/2$ .

and the corresponding descriptor

$$\psi_a^{\rho\theta} = \sum_{p \in [n]^2} w_p M_{\rho\theta} g_{\rho\theta}(\phi_a^p, p) + \mathbf{m}_{\rho\theta}. \quad (12)$$

Different parameterizations, *i.e.* using different coordinate system, provide tolerance to different kinds of misalignment between patches. Cartesian offers tolerance to translation misalignment, while polar offers tolerance to rotation and scale misalignment. To benefit from both types of tolerance, we further use the combined encoding that uses the two coordinate systems and is produced by concatenation of the previous encoding. It is defined as function  $g_c : \mathbb{R}^d \times \mathbb{R}^d \times [n]^2 \rightarrow \mathbb{R}^{2D(2s+1)^2}$  given by

$$g_c(\phi_a^p, \tilde{\phi}_a^p, p) = \left( (\phi_a^p \otimes f(x_p) \otimes f(y_p))^T, (\tilde{\phi}_a^p \otimes f(\rho_p) \otimes f(\theta_p))^T \right)^T \quad (13)$$

where  $\tilde{\phi}$  is used to show that the two encodings do not need to rely on the same FCN  $\phi$ . Subscript  $c$  refers to the combined coordinate system, but we skip  $xy\rho\theta$  to simplify the notation. The final descriptor proposed in this work is

$$\star \psi_a^c = \sum_{p \in [n]^2} w_p M_c g_c(\phi_a^p, \tilde{\phi}_a^p, p) + \mathbf{m}_c \quad (14)$$

where  $M_c \in \mathbb{R}^{D \times 2d(2s+1)^2}$ , and left superscript  $\star$  is used to denote that a separate FCN is used for each encoding, correspondingly coordinate system.

Convolutional part $\phi$			$\psi^{xy}$		$N=\{32, 64\}$	
Conv. layer	Param. matrix shape	# Parameters	HardNet	$N = 32$	$N = 64$	
1	[ 1, 32, 3, 3 ]	288	$\phi$	285,984	285,984	$\phi$
2	[ 32, 32, 3, 3 ]	9,216	FC	1,048,576	4,194,304	$M, \mathbf{m}$
3	[ 32, 64, 3, 3 ]	18,432	Total	1,334,560	4,480,288	Total
4	[ 64, 64, 3, 3 ]	36,864				
5	[ 64, 128, 3, 3 ]	73,728	$\psi^c$	$N=\{32, 64\}$		$\psi^c$
6	[ 128, 128, 3, 3 ]	147,456		$s=1$	$s=2$	
Total		285,984	$\phi$	285,984	285,984	$\phi$
			$M, \mathbf{m}$	295,040	819,328	$M, \mathbf{m}$
			Total	581,024	1,105,312	Total

$\psi^{xy}$		$N=\{32, 64\}$	
		$s=1$	$s=2$
$\phi$	285,984	285,984	285,984
$M, \mathbf{m}$	147,584	409,728	409,728
Total	433,568	695,712	695,712

$\psi^{xy}$		$N=\{32, 64\}$	
		$s=1$	$s=2$
$\phi$	285,984	285,984	285,984
$\tilde{\phi}$	285,984	285,984	285,984
$M, \mathbf{m}$	295,040	819,328	819,328
Total	867,008	1,391,296	1,391,296

Table 1. Number of parameters for different models. The convolutional part  $\phi$  has identical architecture for all models. Cases where both  $\phi$  and  $\tilde{\phi}$  appear use a separate convolutional part for the Cartesian and the polar descriptor. These specifications correspond to  $d = 128$ , and  $D = 128$ . The resulting  $n$  is equal to 8 and 16 for  $N$  equal to 32 and 64, respectively. We report  $M$  and  $\mathbf{m}$  due to limited space, but we refer to  $M_{xy}$ , and  $M_c$  according to the respective table, and similarly for  $\mathbf{m}$ . Descriptor  $\psi^{\rho\theta}$  has identical requirements as descriptor  $\psi^{xy}$ . The parameter requirements of our descriptor remain unchanged for different patch size  $N$ .

## 4. Implementation details

In this section, we provide implementation details that concern the efficiency of the aggregation, describe the different architectures and their required number of parameters, and finally discuss the training procedure.

### 4.1. Efficient aggregation

We describe the implementation details for variant  $\psi_a^{xy}$ , but these hold for other variants too in the same way. Vectors  $w_p f(x_p) \otimes f(y_p) \in \mathbb{R}^{(2s+1)^2}$  that encode positions  $p \in [n]^2$  are fixed for the 2D grid of size  $n \times n$ . Thus, we pre-compute and store them in matrix  $F \in \mathbb{R}^{n^2 \times (2s+1)^2}$ . We reshape 3D tensor  $\phi_a$  into matrix  $\Phi \in \mathbb{R}^{n^2 \times d}$ . Given these two matrices and due to the linearity of matrix to vector multiplication we can re-write the descriptor as

$$\begin{aligned} \psi_a^{xy} &= \sum_{p \in [n]^2} w_p M_{xy} g_{xy}(\phi_a^p, p) + \mathbf{m}_{xy}, \\ &= M_{xy} \left( \sum_{p \in [n]^2} w_p g_{xy}(\phi_a^p, p) \right) + n^2 \mathbf{m}_{xy}, \end{aligned} \quad (15)$$

$$\begin{aligned} &= M_{xy} \left( \sum_{p \in [n]^2} w_p \phi_a^p \otimes f(x_p) \otimes f(y_p) \right) + n^2 \mathbf{m}_{xy}, \\ &= M_{xy} \text{vec}(\Phi^\top F) + n^2 \mathbf{m}_{xy}. \end{aligned} \quad (16)$$

Multiplication  $\Phi^\top F$  makes the computation memory efficient because it avoids explicit storing of the Kronecker

product for each  $p$ . To evaluate (15), the memory requirements are  $n^2 d (2s+1)^2$  numbers, while to evaluate (16), only  $n^2 (d + (2s+1)^2)$  numbers are allocated. Using setup  $d = 128$  and  $s = 2$  in our experiments, the memory requirements are reduced by a factor of 20.9.

### 4.2. Architecture

We use the HardNet+ [30] architecture for the convolution part, since HardNet+ achieves state-of-the-art performance on all benchmarks. We also use it as a baseline to compare with.

The statistics of the convolutional part  $\phi$  are described in Table 1 (left). Each convolutional layer is followed by batch normalization and ReLU, while no bias is used. Table 1 (right) provides the total number of parameters for HardNet+ and our networks, namely, plain polar or Cartesian encoding with different dimensionality of the explicit feature maps ( $s = 1$  and  $s = 2$  frequencies used), and the joint encoding with a common ( $\phi$ ) or separate ( $\phi$  and  $\tilde{\phi}$ ) convolutional part. Note that for the joint encoding with separate convolutional parts and  $s = 2$  frequencies, the proposed network needs roughly the same number of parameters as HardNet+ with input patch of size  $32 \times 32$  pixels ( $N = 32$ ). In all other settings of the proposed architecture, the number of parameters is significantly reduced. Importantly, the number of parameters for larger patch sizes (such as  $64 \times 64$ ), that provide better performance, the number of parameters stays fixed for the proposed architecture. For Hardnet+, the number of parameters of the FC layer increases by a factor of 4 for  $64 \times 64$  input patches.



### 4.3. Training

We would like to highlight the contribution of the explicit spatial encoding and to provide direct comparison to the current state-of-the-art descriptor construction. To avoid changing many things at the same time, we follow exactly the same training procedure as HardNet+, which we briefly review below.

The network is trained with the triplet loss defined as

$$\ell(\hat{\psi}_{an}, \hat{\psi}_{pos}, \hat{\psi}_{neg}) = [1 - ||\hat{\psi}_{an} - \hat{\psi}_{pos}|| + ||\hat{\psi}_{an} - \hat{\psi}_{neg}||]_+, \quad (17)$$

acting on a triplet formed by an anchor, a positive (matching to the anchor), and a negative (non-matching to the anchor) descriptor. A batch of size 1024 patches is constructed from 512 pairs of anchor-positive descriptors. Regarding a particular pair in the batch, the positive descriptors of all other pairs are considered as candidate negatives. Finally, the one with the smallest Euclidean distance to the anchor within the batch is chosen as a hard negative to form a triplet.

We use Stochastic Gradient Descent (SGD) to perform the training. The total training set consists of 2 million anchor-positive pairs and the training lasts 10 epochs. Data augmentation is employed by random patch rotation, scaling and flipping. The learning rate is set to 10, and linearly decays to zero withing 10 epochs. Momentum is equal to 0.9 and weight decay to  $10^{-4}$ . Random orthogonal initialization is used for the weights of the network [42]. The method is implemented in the PyTorch framework.

## 5. Experiments

We first describe the datasets and the evaluation protocols used in our experiments, and then present qualitative results showing the impact of the training on patch similarity. Finally we present the results achieved by different variants of our descriptor and show a comparison with the state of the art.

### 5.1. Datasets and protocols.

We use two publicly available patch datasets, namely *PhotoTourism* (PT) [55] and *HPatches* (HP) [4]. We use the former for both training and evaluation, while the latter only for evaluation when training on PT to show the generalization ability of the descriptor.

The PT dataset consists of following 3 separate sets, Liberty, Notredame and Yosemite. Each consists of local features detected with the Difference-of-Gaussians (DoG) detector and verified through an SfM pipeline. Each set comprises about half a million  $64 \times 64$  patches, associated with a discrete label which is the outcome of SfM verification. The test set consists of 100k pairs of patches corresponding to the same (positive) 3D point, and an equal number corresponding to different (negative) 3D points. The metric used

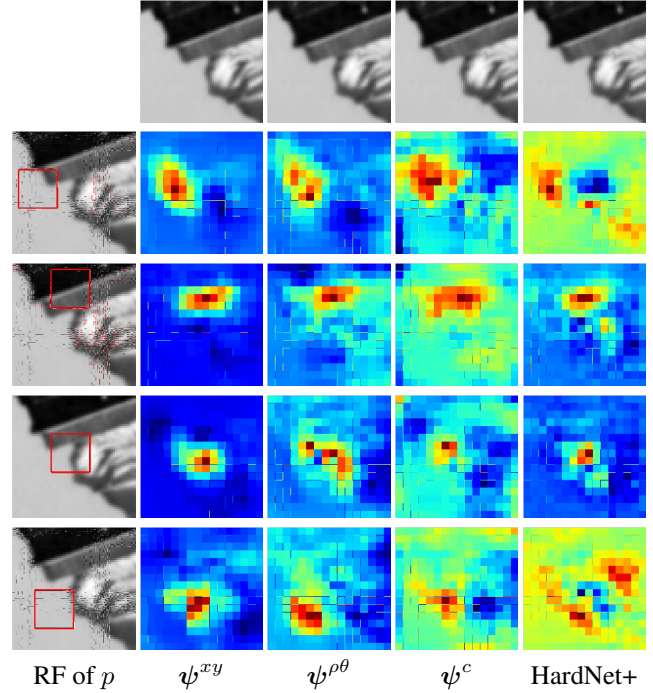


Figure 2. Visualization of similarity between position  $p$  of the  $n \times n$  grid (rows) on a patch and another whole patch for different methods (columns). Heat-maps are normalized to  $[0, 1]$  with red corresponding to the maximum similarity. Red box is used to depict the receptive field (RF) of  $p$ .

to measure performance is the *false positive rate at 95% of recall* (FPR@95). Models are trained on one set and tested on the other two, and the mean of 6 scores is reported.

The HP dataset contains patches of higher diversity and is more realistic. Evaluation is performed on three different tasks, namely verification, retrieval, and matching. Despite the fact that we do not train on HP, we evaluate on all 3 train/test splits and report the average performance to allow future comparisons. We follow the common practice and train our descriptor on Liberty of PT to evaluate on HP.

We repeat each experiment three times, with different random seeds to initialize the parameters, and report mean and standard deviation of the 3 runs. We followed this policy for all variants and datasets.

Recently, larger and more diverse datasets [31, 26] have been introduced to improve local descriptor training. These are shown to improve the performance of state-of-the-art descriptors even by simply replacing the training dataset. We have not included them in our experiments but expect the impact to be similar on our descriptor too.

### 5.2. Visualizing patch similarity.

We construct encodings  $g(\mathbf{v}, p)$ , before aggregation, for our descriptors and for the conventional case and construct a similarity map to analyze the impact of the po-

Test			Liberty		Notredame		Yosemite	
Train	# Parameters	Mean	No	Yo	Li	Yo	Li	No
HardNet+ †	1,334,560	1.51	1.49	2.51	0.53	0.78	1.96	1.84
HardNet+	1,334,560	1.43 ± 0.02	1.25 ± 0.03	2.35 ± 0.03	0.48 ± 0.01	0.74 ± 0.02	2.15 ± 0.01	1.61 ± 0.10
* $\psi^c, s=1$	867,008	1.53 ± 0.03	1.27 ± 0.03	2.31 ± 0.08	0.48 ± 0.02	0.82 ± 0.05	2.58 ± 0.08	1.72 ± 0.09
* $\psi^c, s=2$	1,391,296	1.36 ± 0.01	1.14 ± 0.03	2.16 ± 0.10	0.42 ± 0.01	0.73 ± 0.02	2.18 ± 0.07	1.51 ± 0.12

Table 2. Performance comparison of the proposed descriptors with the state-of-the-art descriptor HardNet+ on the PhotoTourism dataset. Performance is measured via FPR@95. We repeat each experiment/training 3 times and report mean performance and standard deviation. Patch size is  $N = 32$ . †: Reported in the original work.

sition encoding. We present such visualization in Figure 2. We pick position  $p$  and compute similarity  $g_{fc}(\phi_a^p, p)^\top g_{fc}(\phi_b^q, q), \forall q \in [n]^2$ , for the conventional case, and  $(M_c g_c(\phi_a^p, p) + \mathbf{m}_c)^\top (M_c g_c(\phi_b^q, q) + \mathbf{m}_c), \forall q \in [n]^2$  for ours in the case of the combined descriptor. We observe how all architectures, including the conventional one, result in large similarity values near  $p$ .

### 5.3. Results and comparisons.

We train and evaluate different variants of the proposed descriptor. If not otherwise stated, we use input patches of size equal to  $32 \times 32$ , which is the standard practice for deep local descriptors. We further examine the case of  $64 \times 64$  input patches. We always set  $d = 128$  and  $D = 128$ . The dimensionality of the feature maps is controlled by  $s$  which we set equal to 1 or 2 in our experiments.

**Reproducing HardNet+.** Our implementation, training procedure, and training hyper-parameters are based on HardNet+<sup>4</sup>. We reproduce its training and report our own results, proving that our benefit is not an outcome of implementation details. We report both the achieved performed in the original publication and our reproduced ones in all the comparisons.

**Baselines for ablation study.** We train and test the following two baselines to see the impact of the position encoding. First, we train a descriptor that encodes convolutional descriptors in  $\phi_a$  in a translation invariant way, *i.e.* no position encoding at all. It is implemented by spatial sum pooling on  $\phi_a$  and given by

$$\psi_a^{\text{sum}} = \sum_{p \in [n]^2} \psi_a^p. \quad (18)$$

The dimensionality of  $\psi_a^{\text{sum}}$  is equal to  $d$  and not  $D$  in this case. However,  $d = D = 128$ , making this descriptor directly comparable to all others.

Second, we train a descriptor that encodes the spatial information simply by concatenation, *i.e.* vectorization of  $\phi_a$ , which does not provide any tolerance to position misalignments. It is given by

$$\psi_a^{\text{cat}} = \text{vec } \psi_a \quad (19)$$

**Impact of position encoding.** We compare our descriptor with HardNet+ on PT and show results in Table 2. Conceptually it is a comparison between the conventional architecture that uses an FC layer to “feed” the convolutional descriptors to, and our kernel-based approach to explicitly encode the spatial information. Our descriptors (with  $s = 2$ ) slightly outperforms HardNet+ while it has roughly the same number of parameters. Even the variant with fewer parameters ( $s = 1$ ) performs similarly.

A more thorough comparison, examining the impact of the explicit spatial encoding, is performed on HP and presented in Figure 4. Firstly, we evaluate  $\psi^{\text{sum}}$  as part of an ablation study. It is translation invariant that totally discards the spatial information. It does not require additional parameters other than the ones for FCN  $\phi$ . It has significantly lower performance compared to all the other descriptors. We additionally tried including multiplication by matrix  $M_{\text{sum}}$  in (18) and did not notice performance improvements. Descriptor  $\psi^{\text{cat}}$  is another case not requiring additional parameters. It is translation variant in a “rigid” way, whose tolerance to translation misalignment is restricted to the amount that the large receptive field offers. Despite the very large dimensionality, it is not a top performer. Even

<sup>4</sup><https://github.com/DagnyT/hardnet>

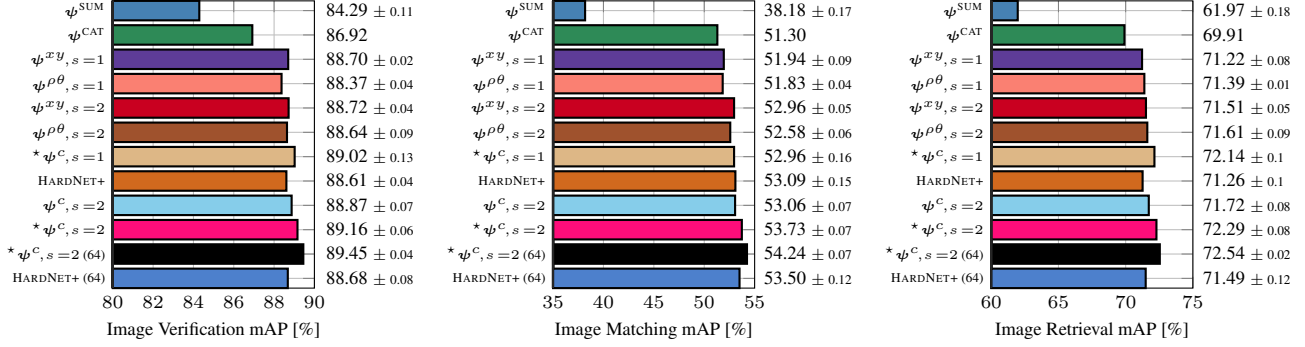


Figure 3. Performance comparison on the HPatches benchmark. The training is performed on the Liberty set of PhotoTourism dataset for all descriptors and with identical setup. Performance is measured via mean Average Precision (mAP). We repeat each experiment/training 3 times and report mean performance and standard deviation (with the exception of  $\psi^{\text{CAT}}$  that due to very high dimensionality was trained only once). All descriptors have 128 dimensions, with the exception of  $\psi^{\text{CAT}}$  which has 8192. The methods are sorted w.r.t. the required number of parameters (top is the least demanding, *i.e.* less parameters). All methods are trained and tested with patch size  $N = 32$  unless when (64) is reported.

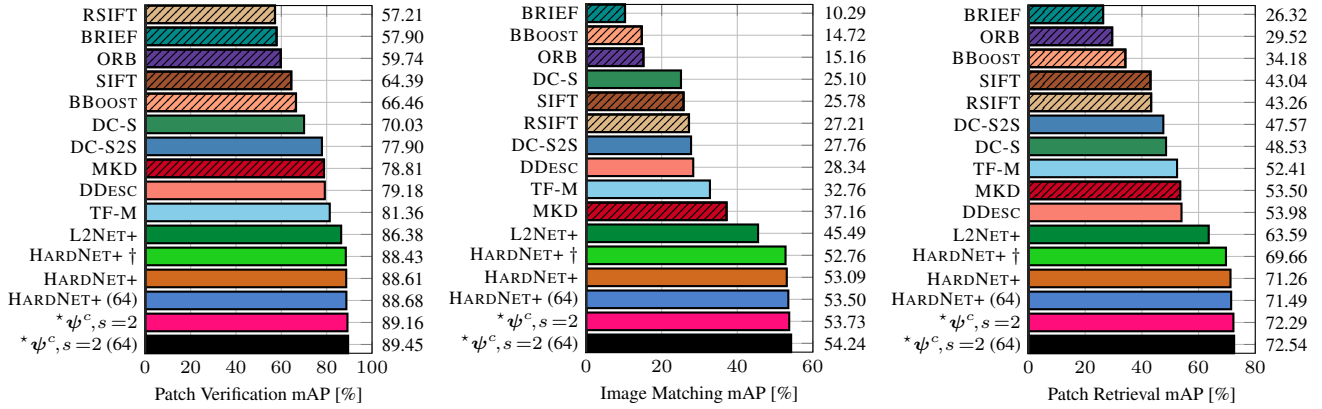


Figure 4. Performance comparison with the state of the art on the HPatches benchmark. The learning for learned descriptors is performed on the Liberty set of PhotoTourism dataset. Hand-crafted descriptors are shown with striped bars. Performance is measured via mean Average Precision (mAP). The performance of our descriptor is the mean of 3 repetitions of each experiment/training. All methods are trained and tested with patch size  $N = 32$  unless when (64) is reported. †: Reported in the original work.

our light-weight variant with as few as 127k additional parameters (excluding  $\phi$ ) recovers most of the performance loss due to lack of spatial information, *i.e.* w.r.t.  $\psi^{\text{SUM}}$ . This result suggests that the common choice of an FC layer for deep local descriptors might be over-parametrized. It is not the best performing either. Our variant  $*\psi^c, s=2$  is consistently the top performing one on all tasks.

**Comparison with the state of the art.** We finally present a comparison to the state of the art on HP in Figure 4. The comparison includes a set of hand-crafted and learned local descriptors, namely RSIFT [2], SIFT [25], BRIEF [12], BBoost [52], ORB [41], MKD [32], Deep-Compare [58], DDesc [46], TFeat [5], L2Net [49] and Hard-Net [30]. The proposed descriptor achieves the best performance with a 128D descriptor on all 3 tasks consistently.

## 6. Conclusions

We interpret conventional convolutional local descriptors as efficient match kernels and show that they learn spatially variant encoding through that last FC layer. We design a novel local descriptor that explicitly encodes the spatial information. We use a combined position parametrization handling different sources of geometric misalignment. It achieves the same performance as state-of-the-art descriptors with fewer parameters and consistently outperforms them on all standard patch benchmarks with the same number of parameters.

**Acknowledgments** This work was supported by the GAČR grant 19-23165S, the OP VVV funded project CZ.02.1.01/0.0/0.0/16.019/0000765 “Research Center for Informatics” and the CTU student grant SGS17/185/OHK3/3T/13.



## References

- [1] Mitsuru Ambai and Yuichi Yoshida. Card: Compact and real-time descriptors. In *ICCV*, 2011. 2
- [2] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 2, 8
- [3] Vassileios Balntas, Edward Johns, Lilian Tang, and Krystian Mikolajczyk. Pn-net: conjoined triple deep network for learning local image descriptors. In *arXiv*, 2016. 2
- [4] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017. 4, 6
- [5] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *BMVC*, 2016. 1, 2, 8
- [6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *CVIU*, 110(3):346–359, 2008. 1, 2
- [7] Liefeng Bo, Kevin Lai, Xiaofeng Ren, and Dieter Fox. Object recognition with hierarchical kernel descriptors. In *CVPR*, 2011. 2
- [8] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *NIPS*, December 2010. 2
- [9] Liefeng Bo and Cristian Sminchisescu. Efficient match kernels between sets of features for visual recognition. In *NIPS*, 2009. 1, 2
- [10] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *CVPR*, volume 1, pages 510–517, 2005. 2
- [11] Andrei Bursuc, Giorgos Tolias, and Hervé Jégou. Kernel local descriptors with implicit rotation matching. In *ICMR*, 2015. 1, 2
- [12] Micheal Calonder, Vincent Lepetit, Cristoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *ECCV*, 2010. 2, 8
- [13] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cedric Bray. Visual categorization with bags of keypoints. In *ECCV Workshop Statistical Learning in Computer Vision*, 2004. 1
- [14] Jingming Dong and Stefano Soatto. Domain-size pooling in local descriptors: Dsp-sift. In *CVPR*, 2015. 1, 2
- [15] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015. 2
- [16] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50, 1988. 1
- [17] Marko Heikkilä, Matti Pietikainen, and Cordelia Schmid. Description of interest regions with local binary patterns. *Pattern recognition*, 42(3):425–436, 2009. 2
- [18] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local descriptors into compact codes. In *IEEE Trans. PAMI*, September 2012. 1
- [19] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. *ECCVW*, 2016. 1
- [20] Yan Ke and Rahul Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *CVPR*, pages 506–513, June 2004. 2
- [21] Theo Gevers Koen van de Sande and Cees Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. PAMI*, 32(9):1582–1596, 2010. 2
- [22] Iasonas Kokkinos and Alan Yuille. Scale invariance without scale selection. In *CVPR*, 2008. 2
- [23] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, 2011. 2
- [24] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *NIPS*, 2018. 1, 2
- [25] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2, 8
- [26] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan. Geodesc: Learning local descriptors by integrating geometry constraints. In *ECCV*, 2018. 2, 3, 6
- [27] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004. 1
- [28] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. PAMI*, 27(10):1615–1630, 2005. 1, 2
- [29] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, F. Schaffalitzky, T. Kadir, and Luc Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005. 1
- [30] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor’s margins: Local descriptor learning loss. In *NIPS*, 2017. 1, 2, 3, 5, 8
- [31] Rahul Mitra, Nehal Doiphode, Utkarsh Gautam, Sanath Narayan, Shuaib Ahmed, Sharat Chandran, and Arjun Jain. A large dataset for improving patch matching. In *arXiv*, 2018. 6
- [32] Arun Mukundan, Giorgos Tolias, Andrei Bursuc, Hervé Jégou, and Ondrej Chum. Understanding and improving kernel local descriptors. *IJCV*, 2019. 1, 2, 3, 4, 8
- [33] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, 2017. 1
- [34] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Semi-convolutional operators for instance segmentation. In *ECCV*, 2018. 1, 2
- [35] Timo Ojala, Matti Pietikainen, and Topi Maenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. PAMI*, 24(7):971–987, 2002. 2
- [36] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. 2

- [37] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, September 2010. 1
- [38] James Philbin, Michael Isard, Josef Sivic, and Andrew Zisserman. Descriptor learning for efficient retrieval. In *ECCV*, 2010. 2
- [39] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. In *IEEE Trans. PAMI*, 2018. 1
- [40] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *ITE Trans. on Media Technology and Applications*, 2016. 1
- [41] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011. 2, 8
- [42] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013. 6
- [43] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. PAMI*, 19(5):530–535, 1997. 2
- [44] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM International Conference on Multimedia*, pages 357–360, 2007. 2
- [45] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *CVPR*, pages 1–8. IEEE, 2007. 2
- [46] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 1, 2, 8
- [47] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Trans. PAMI*, 36(8):1573–1585, 2014. 2
- [48] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 1
- [49] Bin Fan Yurun Tian and Fuchao Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *CVPR*, 2017. 1, 2, 3, 8
- [50] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Trans. PAMI*, 32(5):815–830, 2010. 2
- [51] Giorgos Tolias, Andrei Bursuc, Teddy Furon, and Hervé Jégou. Rotation and translation covariant match kernels for image retrieval. *CVIU*, 140:9–20, 2015. 3
- [52] Tomasz Trzcinski, Mario Christoudias, Vincent Lepetit, and Pascal Fua. Learning image descriptors with the boosting-trick. In *NIPS*, 2012. 8
- [53] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010. 3
- [54] Peng Wang, Jingdong Wang, Gang Zeng, Weiwei Xu, Hongbin Zha, and Shipeng Li. Supervised kernel descriptors for visual recognition. In *CVPR*, 2013. 2
- [55] Simon Winder and Matthew Brown. Learning local image descriptors. In *CVPR*, 2007. 2, 6
- [56] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, pages 467–483. Springer, 2016. 2
- [57] Guoshen Yu and Jean-Michel Morel. A fully affine invariant image comparison method. In *ICASSP*, pages 1597–1600. IEEE, 2009. 2
- [58] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 1, 2, 8