Discriminative Learning of Deep Convolutional Feature Point Descriptors

Edgar Simo-Serra[*,1,5], Eduard Trulls[*,2,5], Luis Ferraz[3]
Iasonas Kokkinos[4], Pascal Fua[2], Francesc Moreno-Noguer[5]
[1] Waseda University, Tokyo, Japan, esimo@aoni.waseda.jp
[2] CVLab, École Polytechnique Fédérale de Lausanne, Switzerland, {eduard.trulls,pascal.fua}@epfl.ch
[3] Catchoom Technologies, Barcelona, Spain, luis.ferraz@catchoom.com
[4] CentraleSupelec and INRIA-Saclay, Chatenay-Malabry, France, iasonas.kokkinos@ecp.fr
[5] Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Barcelona, Spain, {esimo,etrulls,fmoreno}@iri.upc.edu

Abstract

Deep learning has revolutionalized image-level tasks such as classification, but patch-level tasks, such as correspondence, still rely on hand-crafted features, e.g. SIFT. In this paper we use Convolutional Neural Networks (CNNs) to learn discriminant patch representations and in particular train a Siamese network with pairs of (non-)corresponding patches. We deal with the large number of potential pairs with the combination of a stochastic sampling of the training set and an aggressive mining strategy biased towards patches that are hard to classify.

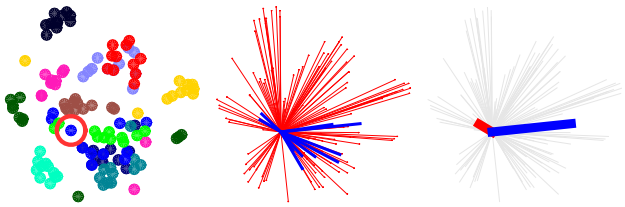
By using the L_2 distance during both training and testing we develop 128-D descriptors whose euclidean distances reflect patch similarity, and which can be used as a drop-in replacement for any task involving SIFT. We demonstrate consistent performance gains over the state of the art, and generalize well against scaling and rotation, perspective transformation, non-rigid deformation, and illumination changes. Our descriptors are efficient to compute and amenable to modern GPUs, and are publicly available.

1. Introduction

Representing local image patches in an invariant and discriminative manner is a major research topic in computer vision. While most descriptors, such as SIFT [16], rely on hand-crafted features [1, 13, 16, 22, 27, 28, 32], there has recently been interest in using machine learning algorithms to learn them from large datasets [20, 23, 29].

In this paper we draw inspiration from the recent success of Deep CNNs in large-scale image classification problems [14, 26] to build discriminative descriptors for local patches. In our case discriminative training does not rely on labels of individual patches, but rather on pairs of corresponding, or non-corresponding patches. For this we use a Siamese network architecture [2] that employs two CNNs with identical parameters to compare pairs of patches; treating the CNN outputs as patch descriptors, we minimize a loss that enforces the L_2 norm of their difference to be small for corresponding patches and large otherwise.

To train this network we rely on the multi-view stereo dataset (MVS) [3], which contains over 1.5M grayscale 64×64 image patches from different views of 500K 3D points. The difficulty with such a large dataset is that it becomes impossible to exhaustively explore all correspond-



(a) Data (b) All pairs (c) "Hard" pairs

Figure 1: To train models with Siamese networks, we need pairs of corresponding and non-corresponding samples. (a) We use t-SNE [30] to display ~100 64×64 patches of 12 3D points from different images (see Fig. 3 for examples). Corresponding patches are drawn with the same color. (b) We single out the red-circled patch, belonging to the blue point cloud, and consider all of its potential pairings. The line length encodes the closeness between this patch and the rest: positive matches in blue, negative in red. Most pairs are easy to discriminate and ineffectual for training. (c) We mine the samples to obtain the closest negative (shortest red line) and the most distant positive (longest blue line). This simple strategy allows us to train discriminative networks over large datasets.

∗ First two authors contributed equally.

This work was partly funded by the Spanish MINECO project RobInstruct TIN2014-58178-R, by the ERA-Net Chistera project ViSen PCIN-2013-047, by EU projects AEROARMS H2020-ICT-2014-1-644271, ISUPPORT H2020-ICT-2014-1-643666 and MOBOT FP7-ICT-2011-600796, and by the ERC project MicroNano.

ing and non-corresponding pairs, so we must resort to some form of random sampling. Based on the observation that after a certain point of learning most pairs are correctly classified, and using them no longer improves the learned embedding, we propose a strategy of aggressive mining of "hard" positives and negatives. During the learning stage we enforce the back-propagation of samples with a large loss, i.e. both corresponding pairs that match poorly and non-corresponding pairs that are hard to discriminate. This proves to be most useful for efficiently learning discriminative descriptors.

We perform in-depth comparisons against both traditional, hand-crafted descriptors [16, 27, 22] as well as learned, state-of-the-art descriptors [23, 29], using Precision-Recall (PR) and its area under the curve (AUC) as a metric, and demonstrate consistent gains in performance. Our descriptors also generalize very well to applications for which they were not specifically trained, demonstrating remarkable robustness against scaling, rotation, viewpoint changes, non-rigid deformations, and varying illumination.

In all of our experiments we use the L_2 distance to compare descriptors, rather than some nonlinear, task-specific metric, as e.g. in [10, 34]. This demonstrates that our descriptors can be used as a drop-in replacement for popular representations such as SIFT, in a manner that is agnostic to the application. Furthermore, as our descriptors are primarily built from convolutions they are very efficient to compute and can be easily parallelized, taking advantage of modern GPUs to greatly speed up their extraction. Our implementation is based on Torch7 [5]. Our feature extraction code and pre-trained models are available from https://github.com/etrulls/deepdesc-release.

2. Related Work

Local features have proven very successful at matching points across images, and are nearly ubiquitous in modern computer vision, with a broad range of applications encompassing stereo, structure from motion, pose estimation, classification, detection, medical imaging, and many others. Recent developments in the design of local image descriptors are moving away from carefully-engineered features [1, 16, 27] and towards learning features from large volumes of data. This line of works includes unsupervised techniques based on hashing as well as supervised approaches using Linear Discriminant Analysis [3, 9, 24], boosting [29], and convex optimization [23].

In this paper we explore solutions based on deep convolutional neural networks (CNNs), which currently are the dominant paradigm in tasks involving semantic information, e.g. image classification [14, 26] or semantic segmentation [15, 4]. Even though it may be unclear whether CNNs are equally appropriate for patch-level applications where semantic information may be missing, we argue that for our particular problem this is indeed the case.

Descriptor learning using CNNs was addressed early in [11, 19], but the experimental results in these works left open questions regarding several practical aspects, such as the most appropriate network architectures and application-dependent training schemes. More recently, the use of Siamese networks for descriptor learning was exploited by concurrent works on joint descriptor and metric learning [10, 33, 34]. Han et al. [10] use a deep convolutional network in a Siamese architecture followed by a fully-connected network that learns a comparison function. Zagoruyko et al. [33] rely on a similar architecture but add a network that only focuses on the center of the image, which they show increases performance, at a computational cost. Zbontar & LeCun [34] trained CNNs for narrow-baseline stereo and obtained the top results on the KITTI benchmark. These approaches rely on larger networks and do not necessarily learn compact, discriminative representations, like ours. In contrast, we show how to exploit discriminative training strategies to build small but powerful models.

One key distinction between [10, 33] and our work is that we aim at using the CNN outputs of our Siamese networks as *direct counterparts to traditional descriptors*—namely, unlike [10, 33, 34] there is no non-linear 'metric network' following the Siamese network application, but rather *we simply use the L_2 distance to compare patches*. In [33] a limited evaluation of L_2-based similarity shows promising results, which however is not entirely clearly outperforming [23]—instead we show substantial gains, which can be also attributed to using the L_2 distance during training. Using descriptors that can be compared with the L_2 distance facilitates the use of efficient methods for nearest neighbor computations, such as KD-trees, which we believe opens up the path to large-scale retrieval applications.

Another deviation of our work from common practice is that we observe that during descriptor training the majority of non-corresponding patch pairs eventually become easy to discern, which stalls the learning of discriminative models. Mining hard negatives is a well-known procedure in the context of sliding-window detectors [8], where the number of negative samples (windows) is virtually unlimited and yet most negatives are easily discriminated once we have already used a certain number of negative samples for training. In this paper we demonstrate that aggressive mining of both "hard" positive and negative samples greatly enhances the learning process: as we detail in the following section, we sample a large number of matches and use the subset with the largest loss to update the network.

3. Learning Deep Descriptors

Given an intensity patch $\mathbf{x} \in \mathbb{R}^d$, the descriptor of $\mathbf{x}$ is a non-linear mapping $D(\mathbf{x})$ that is expected to be discriminative, i.e. descriptors for image patches corresponding to the

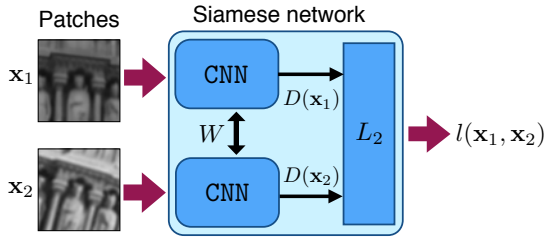


Figure 2: Schematic of a Siamese network, where pairs of input patches are processed by two copies of the same CNN.

same point should be similar, and dissimilar otherwise.

We propose to learn such descriptors with a Siamese network [2], where a nonlinear mapping is represented by a CNN that is optimized for pairs of corresponding or non-corresponding patches, as shown in Fig. 2. We propagate the patches through the model to extract the descriptors and then compute their L_2 norm, which is a standard similarity measure for image descriptors. The objective is to learn a descriptor that places non-corresponding patches far apart and corresponding patches close together.

In the context of multiple-view geometry, descriptors are typically computed for salient points where scale and orientation can be reliably estimated, for invariance. Patches then capture local projections of 3D scenes. Let us consider that each image patch $\mathbf{x}_i$ has an index p_i that uniquely identifies the 3D point which roughly projects onto the 2D patch, from a specific viewpoint. Using the L_2 norm as a similarity metric between descriptors we write our objective in terms of the hinge embedding loss [18]:

$$l(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \|D(\mathbf{x}_1) - D(\mathbf{x}_2)\|_2, & p_1 = p_2 \\ \max(0, C - \|D(\mathbf{x}_1) - D(\mathbf{x}_2)\|_2), & p_1 \neq p_2 \end{cases}$$

where p_1, p_2 are the indices of the 3D points projecting to $\mathbf{x}_1, \mathbf{x}_2$ respectively. This loss penalizes corresponding pairs that are placed far apart, and non-corresponding pairs that are less than C units apart—in particular, when $\|D(\mathbf{x}_1) - D(\mathbf{x}_2)\|_2 = 0$ we pay the maximal cost, C, and as their distance increases the loss eventually reaches zero.

3.1. CNN-based Descriptors

When designing the structure of the CNN we are limited by the size of the input data: in our case 64×64 patches, from the MVS dataset [3], while we extract descriptors of the same size as SIFT [16], i.e. 128 dimensions. Note that larger patches and/or output spaces would allow us to consider possibly more informative descriptors, but at the same time they would be also more susceptible to occlusions and slower to train and compute.

We explored many configurations regarding the number of filters, hidden units, mapping, normalization, and pooling. Such architectures are detailed in the supplemental material, but due to space constraints we use only our top performing model, i.e. 'CNN3', for the following experiments.

The architecture of this three-layer network is detailed in Table 1. Each convolutional layer consists of four sublayers: filter layer, non-linearity layer, pooling layer and normalization layer. Since sparser connectivity has been shown to improve performance while lowering parameters and increasing speed [6], except for the first layer, the filters are not densely connected to the previous layers. Instead, they are sparsely connected at random, so that the mean number of connections each input layer has is constant. Each filter of the second and third layer are also connected randomly to 8 feature maps of the previous layer so that the mean number of connections stays roughly equal to 16 connections per filter output.

Layer	1	2	3
Input size	64 × 64	29 × 29	8 × 8
Filter size	7 × 7	6 × 6	5 × 5
Output channels	32	64	128
Pooling & Norm.tion	2 × 2	3 × 3	4 × 4
Nonlinearity	Tanh	Tanh	Tanh
Stride	2	3	4

Table 1: Architecture of the proposed three-layer network: a 64 × 64 input yields a 128-dimensional output in layer 3.

Regarding the non-linear layer, we use hyperbolic tangent units (Tanh), as we found it to perform better than Rectified Linear Units (ReLU). We use L_2 pooling for the pooling sublayers, which has been shown to outperfom the more standard max pooling [21]. Normalization is also important for deep networks [12] and paramount for descriptors [17]. We use subtractive normalization, i.e. subtract the weighted average over a 5×5 neighbourhood with a Gaussian kernel after the first and second layers.

3.2. Stochastic Sampling Strategy and Mining

Our goal is to optimize the network parameters from an arbitrarily large set of training patches. Let us consider a dataset with k patches and $m \leq k$ unique 3D patch indices, each with c_i corresponding image patches. Then, the number of matching image patches, $\mathcal{P}$ (positives) and the number of non-matching images patches, $\mathcal{N}$ (negatives) is:

$$\mathcal{P} = \sum_{i=1}^{m} \frac{c_i(c_i - 1)}{2} \quad \text{and} \quad \mathcal{N} = \sum_{i=1}^{m} c_i \, (k - c_i) \, . \qquad (1)$$

Since both $\mathcal{P}$ and $\mathcal{N}$ are intractably large, we resort to Stochastic Gradient Descent, using random subsets of our training set to estimate the gradient of our loss function. For positives we can randomly sample a set of s_p 3D point indices from the set $\{p_1, \ldots, p_m\}$, and for each chosen 3D index p_i we randomly pick two 2D patches with corresponding 3D point indices.

For negatives one simple idea would be to randomly choose s_n random pairs with non-matching indices; but

once the network has reached a reasonable level of performance, most non-corresponding points will already have a distance above C, contributing nothing to the loss—and the gradient. This can result in a very small and noisy estimate of the gradient, effectively stalling the learning process.

Instead, we iterate over non-corresponding patch pairs to search for "hard" negatives, i.e. pairs that are close in descriptor space and incur a high loss. In this manner it becomes feasible to train discriminative models faster while also increasing performance.

In particular, at each epoch we generate a set of s_n randomly chosen patch pairs, and after forward-propagation through the network and computing their loss we keep only a subset of the s_n^H "hardest" negatives, which are back-propagated through the network in order to update the weights. Additionally, the same procedure can be used over the positive samples, i.e. we can sample s_p corresponding patch pairs and prune them down to the s_p^H "hardest" positives. Our experimental results clearly show that the combination of aggressive mining for both positive and negative patch pairs allows us to greatly improve the discriminative capability of our learned descriptors.

4. Results

For training we use the Multi-view Stereo Correspondence dataset (MVS) [3], which consists of 64×64 grayscale image patches sampled from 3D reconstructions of the Statue of Liberty (LY), Notre Dame (ND) and Half Dome in Yosemite (YO). Patches are extracted using the Difference of Gaussians detector [16], and determined as a valid correspondence if they are within 5 pixels in position, 0.25 octaves in scale and $\pi/8$ radians in angle. Fig. 3 shows some samples from each set, which contain significant changes in position, rotation and illumination conditions, and often exhibit very noticeable perspective changes.

We join the data from LY and YO to form a training set with over a million patches. Out of these we reserve a subset of 10,000 unique 3D points for validation (∼30,000 patches). The resulting training set contains 1,133,525 possible positive combinations and 1.117×10^{12} possible negative combinations. This skew is common in correspondence problems such as stereo or structure from motion—we address it with aggressive mining. We use this split to evaluate different architectures and configurations, and then train the top-performing model over the two remaining splits.

A popular metric for classification systems is the Receiving Operator Characteristic (ROC), used e.g. in [3], which can be summarized by its Area Under the Curve (AUC). However, ROC curves can be misleading when the number of positive and negative samples are very different [7], and are already nearly saturated for the SIFT baseline. A richer performance indicator is the Precision-Recall curve (PR). We benchmark our models with PR curves and their AUC. In particular, we simulate the 'needle in a haystack' setting of retrieval by having a thousandfold more negative than positive pairs: for each of the 10,000 unique points in our validation set we generate a single positive pair, by randomly sampling two corresponding patches, and 1,000 non-corresponding patches, chosen from the remaining points.

Results outline: We explored multiple architectures and configurations—some of these results were omitted from the paper due to space constraints, but they remain available in the supplemental material. We study the effect of mining for "hard" samples in Sec. 4.2. We then evaluate our top-performing models over the test set in Sec. 4.3. To build a test set we follow the same procedure as for validation, evaluating 10,000 points with 1,000 negatives each, over 10 different folds (see Sec. 4.3 for details). We consider four splits: LY+YO (tested on ND), LY+ND (tested on YO), and YO+ND (tested on LY), plus a final split with training data from all three sets.

Finally, we apply the models learned over the MVS dataset to different applications. In Sec. 4.4 we study the robustness of our descriptors to patch rotation. In Sec. 4.5 we use our models to match wide-baseline images from a different stereo dataset. In Sec. 4.6 we benchmark our descriptors on a recent dataset with very challenging non-rigid deformations and drastic changes in illumination. Our models outperform state-of-the-art baselines in every case, without fine-tuning over new data, and over considerably different application domains.

4.1. Network training

We use Stochastic Gradient Descent with a learning rate of 0.01 that decreases by a factor of 10 every 10,000 iterations, and a momentum of 0.9, to accelerate learning. Following common practice, we preprocess the patches using mean and standard deviation normalization. We use a subset of the data for validation and stop training when the network evaluation metric converges. Apparently due to the large pool of positives and negatives available for training and the relatively small number of parameters of our architectures, we did not encounter overfitting problems.

4.2. Mining

We analyze the effect of both positive and negative mining by training different models in which a large, initial pool of s_p positives and s_n negatives are pruned down to a smaller number of "hard" positive and negative matches, which are used to update the parameters of the network. We observe that increasing the batch size does not offer benefits in training: see Table 2. We thus keep the batch size fixed to $s_n^H = 128$ and $s_p^H = 128$, and increase the ratio of both negative mining $r_n = s_n/s_n^H$ and positive mining $r_p = s_p/s_p^H$. We keep all other parameters constant. In the following, we use the notation r_p/r_n, for brevity.

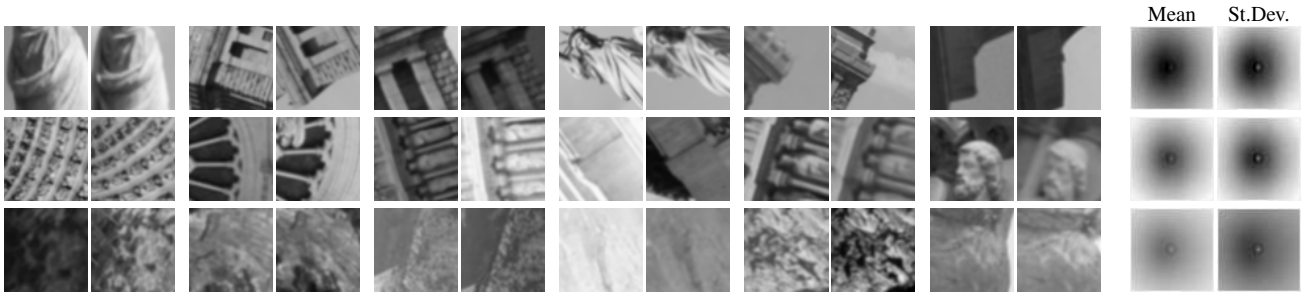


Figure 3: Pairs of corresponding samples from the MVS dataset. Top: 'Liberty' (LY). Middle: 'Notre Dame' (ND). Bottom: 'Yosemite' (YO). Right: we compute the pixel difference between corresponding patches on each set and show their mean/std.

s_p	s_n	r_p	r_n	Cost	PR AUC
128	128	1	1	—	0.366
256	256	1	1	—	0.374
512	512	1	1	—	0.369
1024	1024	1	1	—	0.325
128	256	1	2	20%	0.558
256	256	2	2	35%	0.596
512	512	4	4	48%	0.703
1024	1024	8	8	67%	**0.746**
2048	2048	16	16	80%	0.538

Table 2: Four top rows: effect of increasing batch size, without mining. Four bottom rows: with mining. Mining factors indicate the samples considered (s_p, s_n), the hardest 128 of which are used for training. Column 5 indicates the fraction of the computational cost spent mining hard samples. These experiments correspond to the validation set.

Large mining factors have a high computational cost, up to 80% of the total computational cost, which includes mining (i.e. forward propagation of all s_p and s_n samples) and learning (i.e. backpropagating the "hard" positive and negative samples). Note that this is only applicable to the learning stage—once the model is deployed, we discard the Siamese network and do not incur the computational costs related to mining. In order to speed up the learning process we initialize the CNN3 models with positive mining, i.e. 2/2, 4/4, 8/8 and 16/16, with an early iteration of a model trained only with negative mining (1/2).

Results are shown in Table 2. We see that for this particular problem, aggressive "hard" mining is fundamental. This is due to the extremely large number of both negatives and positives in the dataset, in combination with models with a relatively low number of parameters. We observe a drastic increase in performance up to 8/8 mining factors.

4.3. Generalization & comparison to state of the art

In this section we consider the three splits for the MVS dataset of [3]. We train the top-performing model (i.e. CNN3), with different mining ratios (1/2, 2/2, 4/4 and 8/8), on a combination of two sets, and test it on the remaining set. We select the training iteration that performs best over the corresponding validation set. The test datasets are very large (up to 633K patches) and we use the same procedure as for validation: we consider 10,000 unique points, each with 1,000 random non-corresponding matches. We repeat this process over 10 folds, thus considering 100,000 sets of one corresponding patch vs 1,000 non-corresponding patches. We show results in terms of PR AUC in Table 3, and the corresponding PR curves are pictured in Fig. 4.

We report consistent improvements over SIFT, a handcrafted descriptor which nevertheless remains the most popular among its brethren. Performance varies significantly from split to split; this is due to the nature of the different sets. 'Yosemite' contains mostly frontoparallel translations with illumination changes and no occlusions (Fig. 3, row 3); SIFT performs well on this type of data. Our learned descriptors outperform SIFT on the high-recall regime (over 20% of the samples; see Fig. 4), and is 28% better overall in terms of PR AUC. The effect is much more dramatic on 'Notredame' and 'Liberty', which contain significant patch translation and rotation, as well as viewpoint changes around outcropping, non-convex objects, which result in occlusions (Fig. 3, rows 1-2). Our learned descriptors outperform SIFT by 91% and 169% over ND and LY, respectively.

Additionally, we pit our approach against the state of the art descriptors of [29] and [23]. For [29] we consider 4 binary descriptor variants (BGM, BinBoost-64, BinBoost-128, and BinBoost-256) and a floating-point variant (L-BGM); for the binary descriptors we use the Hamming distance, instead of the Euclidean distance. For VGG [23] we re-train their models over two sets at a time, to provide a fair comparison with ours. We consider only their top-performing variant, i.e. the largest descriptor. The VGG descriptor considers multiple compression settings—we show the results for the best model (i.e. floating point, size 80).

The results are summarized in Table 4 and shown in Fig. 5. Due to the binary nature of the Hamming distance, the curves for the binary descriptors can be seen to have a sawtooth shape where each tooth corresponds to a 1-bit difference. Our approach outperforms the baselines on 'Notredame' and 'Liberty'. On 'Yosemite' VGG obtains the best results, and our approach outperforms the

Train	Test	SIFT	CNN3 mine-1/2	CNN3 mine-2/2	CNN3 mine-4/4	CNN3 mine-8/8
LY+YO	ND	0.349	0.535	0.555	0.630	**0.667**
LY+ND	YO	0.425	0.383	0.390	0.502	**0.545**
YO+ND	LY	0.226	0.460	0.483	0.564	**0.608**

Table 3: PR AUC for the generalized results over the three MVS dataset splits, for different mining factors.

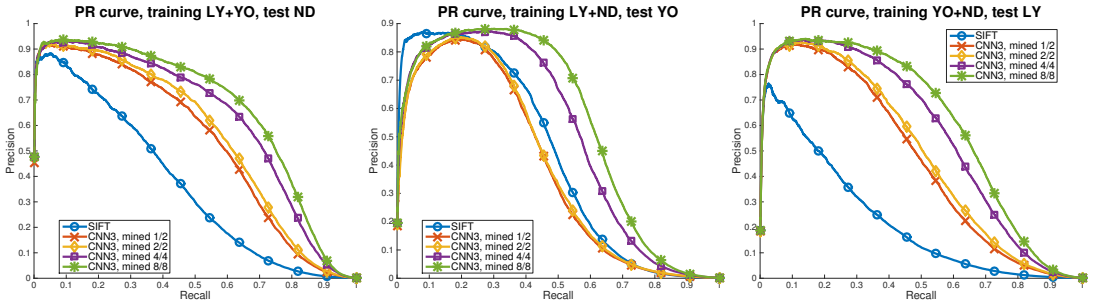


Figure 4: PR curves for the generalized results over the three MVS dataset splits, for different mining factors.

Test	SIFT (128f)	BGM (256b)	L-BGM (64f)	BinBoost-{64,128,256} (64b)	(128b)	(256b)	VGG (80f)	Ours (128f)
ND	0.349	0.487	0.495	0.267	0.451	0.549	0.663	**0.667**
YO	0.425	0.495	0.517	0.283	0.457	0.533	**0.709**	0.545
LY	0.226	0.268	0.355	0.202	0.346	0.410	0.558	**0.608**
All	0.370	0.440	0.508	0.291	0.469	0.550	0.693	**0.756**

Table 4: Generalized results: PR AUC over the three MVS dataset splits, and a new split with data from all three sets, against SIFT, BinBoost [29], and VGG [23]. We re-train VGG with data from two sets (rows 1-3) and all sets (row 4).

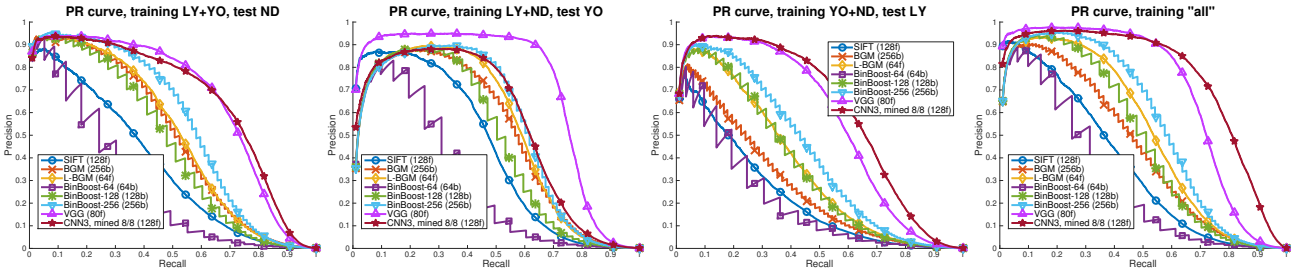


Figure 5: Generalized results: PR curves over the three MVS splits, and a new split with data from all three sets, compared to SIFT, Binboost [29], and VGG [23]. We re-train VGG with data from two sets (columns 1-3) and all sets (column 4).

other baselines by a smaller margin. We argue that this is due to the fact that ND/LY are not representative of YO. We illustrate this in Fig. 3 (right), where we compute the pixel difference over every *corresponding* pair of patches in each set, and plot its mean and std. deviation: YO exhibits a much smoother mean and a smaller variance, which corresponds with our observation that unlike ND/LY, it contains mostly lighting changes and small displacements. This hurts our approach more than VGG, which builds on traditional grid-based descriptors [23]. To illustrate this point, we re-train both our models and VGG [23] over a new split ('All') with data from all three sets, following the methodology of Sec. 4. The results in Fig. 5 (right) and in the last row of Table 4 show a 9.1% relative improvement over VGG.

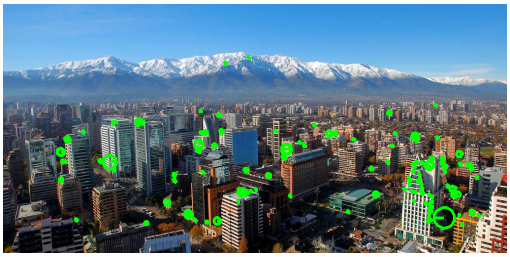
Finally, we provide the computational cost in Table 5. The CPU descriptors run on a 12-core 3.47GHz Xeon CPU, multi-threaded. Our GPU variant runs on a Titan Black. SIFT and VGG rely on VLFeat [31], while our approach can still be optimized, particularly for dense computation.

4.4. Robustness to Rotation

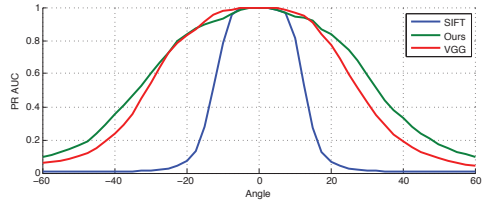
Robustness to rotation is crucial to many applications, as most rotation-invariant detectors can incur in significant errors when estimating the orientation of a patch. For this

	Ours (GPU)	Ours (CPU)	SIFT	VGG [23]
Time (ms)	0.76	4.81	0.14	4.21

Table 5: Computational cost for one descriptor (in batch).



(a) Feature points.



(b) PR AUC results under increasing rotation.

	SIFT	VGG [23]	Ours
Area under the curve	0.223	0.507	**0.564**

(c) Area under the curve of (b).

Figure 6: Robustness to Rotation.

purpose we evaluate the performance of our descriptor under rotation errors, in a synthetic scenario. To do this we extract keypoints with a Difference of Gaussians detector, and extract their correspondent descriptors. We then increase the rotation of each patch in a systematic manner, and compute descriptors for new features. We match the descriptors and calculate the PR AUC, for increasing values of the rotation error. We evaluate SIFT and the learned, state-of-the-art VGG descriptor [23] in addition to ours, and show results in Fig. 6. In particular we use an image of Santiago de Chile and randomly extract 147 patches (shown in Fig. 6-(a)), constrained to the center of the image to avoid border artefacts. We observe that while all descriptors perform well below 10 degrees of rotation, SIFT's performance begins to deteriorate by that point. Our descriptor proves the most robust in this scenario, with a 11.2% relative improvement over VGG, using the top-performing model in either case. This robustness against rotation is particularly valuable when computing dense descriptors, where rotating each patch independently would incur in a considerable computational overhead.

4.5. Wide-baseline matching

In this section we apply our models to the wide-baseline stereo dataset of [25], which consists of two multi-view sets of high-resolution images with ground truth depth maps.

Descriptor	Training	8×8	16×16	24×24	32×32	48×48	64×64
Ours	LY+YO	0.743	**0.912**	0.915	0.916	0.918	**0.923**†
Ours	LY+ND	0.627	0.910	**0.917**	0.916	0.912	0.919
Ours	YO+ND	0.754	0.911	**0.917**	**0.921**	**0.922**	0.922
VGG [23]	YO	0.597	0.850	0.876	0.889	0.897	0.894
VGG [23]	ND	0.598	0.840	0.872	0.877	0.891	0.880
VGG [23]	LY	0.586	0.839	0.875	0.874	0.887	0.879
Daisy [27]	–	**0.796**	0.875	0.878	0.873	0.862	0.835
SIFT [16]	–	0.677	0.837	0.846	0.841	0.798	0.772

Table 6: Stereo matching, baseline '3' vs '4'.

Descriptor	Training	8×8	16×16	24×24	32×32	48×48	64×64
Ours	LY+YO	0.481	**0.763**	0.762	0.755	0.713	**0.690**
Ours	LY+ND	0.368	0.757	**0.780**†	0.765	0.703	0.677
Ours	YO+ND	0.504	0.759	0.770	**0.777**	**0.716**	0.685
VGG [23]	YO	0.338	0.633	0.669	0.687	0.672	0.632
VGG [23]	ND	0.330	0.617	0.641	0.657	0.628	0.590
VGG [23]	LY	0.316	0.604	0.641	0.660	0.630	0.582
Daisy [27]	–	**0.526**	0.719	0.735	0.714	0.660	0.594
SIFT [16]	–	0.357	0.551	0.563	0.587	0.540	0.532

Table 7: Stereo matching, baseline '3' vs '5'.

This allows us to further evaluate the generality of our models across different datasets, and to study how robust the descriptors are against perspective transformations.

We pit our descriptor against SIFT, Daisy [27] and VGG [23]. We consider the 'fountain' set, which contains much wider baselines in terms of angular variation and provides a harder challenge. Fig. 7 (top) shows the images used—we match '3' (the rightmost view) against '4'-'8'. We sample 1000 (non-occluded) points randomly and use the ground truth depth maps to determine their correspondence over the opposite camera. We match every point in one camera with every possible correspondence, and compute PR curves. The difference in viewpoint across increasing baselines creates perpective transformations, which include scaling, rotation, and partial occlusions. We explore different patch sizes, from 8×8 up to 64×64. Note that our models were trained with patches of size 64×64, and we upscale the patches if required; we expect that better performance can be obtained by training filters of a size commensurate to the patch. The results are shown in Tables 6-10; the top perfomer for every setting is highlighted in bold, and the top performer for a given baseline is marked with †. As expected, large patches are more informative across narrow baselines, whereas small patches perform better across wide baselines. Our descriptors outperform the baselines in just about every scenario, proving that they generalize well across datasets. Note that both our models and VGG are trained with the MVS dataset [3].

4.6. Deformation and Varying Illumination Dataset

Lastly, we evaluate our descriptors on a recent, publicly available dataset featuring challenging non-rigid deformations and very severe illumination changes [22]. The dataset consists of a series of photographs of 12 deformable ob-

Figure 7: Samples from the experiments of Sec. 4.5 (top, dataset from [25]) and Sec. 4.6 (bottom, dataset from [22]).

Descriptor	Training	8×8	16×16	24×24	32×32	48×48	64×64
Ours	LY+YO	**0.283**	**0.575**†	**0.564**	0.540	0.478	**0.456**
Ours	LY+ND	0.181	0.543	0.561	0.543	0.468	0.424
Ours	YO+ND	0.271	0.547	0.561	**0.556**	**0.490**	0.439
VGG [23]	YO	0.232	0.414	0.466	0.456	0.420	0.400
VGG [23]	ND	0.234	0.402	0.441	0.440	0.381	0.372
VGG [23]	LY	0.223	0.389	0.424	0.423	0.388	0.365
Daisy [27]	–	0.278	0.482	0.510	0.500	0.440	0.363
SIFT [16]	–	0.143	0.340	0.328	0.333	0.300	0.308

Table 8: Stereo matching, baseline ‘3’ vs ‘6’.

Descriptor	Training	8×8	16×16	24×24	32×32	48×48	64×64
Ours	LY+YO	**0.138**	0.337	0.331	0.301	0.240	0.218
Ours	LY+ND	0.088	0.319	**0.336**	0.339	0.253	0.197
Ours	YO+ND	0.121	**0.341**†	0.333	**0.340**	**0.275**	**0.228**
VGG [23]	YO	0.109	0.226	0.250	0.239	0.220	0.174
VGG [23]	ND	0.115	0.229	0.242	0.228	0.198	0.182
VGG [23]	LY	0.107	0.215	0.233	0.220	0.192	0.166
Daisy [27]	–	0.131	0.283	0.323	0.315	0.252	0.172
SIFT [16]	–	0.066	0.158	0.149	0.152	0.125	0.138

Table 9: Stereo matching, baseline ‘3’ vs ‘7’.

Descriptor	Training	8×8	16×16	24×24	32×32	48×48	64×64
Ours	LY+YO	**0.080**	**0.188**†	0.180	0.156	**0.110**	**0.088**
Ours	LY+ND	0.058	0.173	0.158	0.153	0.087	0.058
Ours	YO+ND	0.078	0.178	**0.183**	**0.159**	0.107	0.082
VGG [23]	YO	0.062	0.125	0.107	0.086	0.080	0.067
VGG [23]	ND	0.062	0.121	0.100	0.075	0.083	0.068
VGG [23]	LY	0.062	0.107	0.094	0.076	0.083	0.064
Daisy [27]	–	0.049	0.098	0.113	0.104	0.060	0.032
SIFT [16]	–	0.028	0.051	0.049	0.045	0.044	0.053

Table 10: Stereo matching, baseline ‘3’ vs ‘8’.

jects, such as clothes and newspapers, which are subjected to four different deformation levels and four different illumination levels, i.e. 16 images per object, for a total of 192 grayscale 640×480 images. Feature points, extracted with Difference-of-Gaussians detectors, are provided for each image. Some examples of the kind of transformations featured in this dataset are shown in Fig. 7 (bottom).

We pit our descriptor against DaLI, SIFT, Daisy and the VGG descriptor, and show the results in Table 11. We evaluate our model trained on three different splits of the MVS dataset, and observe that they all obtain similar performance. We outperform the current state of the art in the deformation (Def.) and deformation with illumination (Def.+Ill.) settings. This is despite having to upscale the image patches from 41×41 pixels to 64×64 pixels, the fact that the image patches are cropped to be circular while our descriptor relies on square patches, and that we trained our descriptors on datasets of rigid, non-deformable objects. In the case of only illumination changes (Ill.), we obtain a performance very close to the DaLI descriptor [22], explicitly designed to deal with these kind of transformations. We also compare favorably to the VGG descriptor [23], which we outperform in every scenario.

Descriptor	Training	Def.	Ill.	Def.+Ill.
Ours	LY+YO	76.568	88.434	75.933
Ours	LY+ND	75.702	87.521	75.606
Ours	YO+ND	**76.731**	88.898	**76.591**
VGG [23]	YO	74.120	87.342	74.765
VGG [23]	ND	72.629	84.690	72.599
VGG [23]	LY	72.602	84.848	72.565
DaLI [22]	-	70.577	**89.895**	72.912
Daisy [27]	-	67.373	75.402	66.197
SIFT [16]	-	55.822	60.760	53.431

Table 11: Results on the dataset of [22]. We evaluate over three different settings, corresponding to deformation changes only (Def.), illumination changes only (Ill.), and both simultaneously (Def.+Ill.). We show the mean accuracy of descriptor matches and highlight the top-performing descriptor for each of setting, in bold.

5. Conclusions

We use Siamese networks to train deep convolutional models for the extraction of image descriptors. Training such models involves small patches, which constraints the network size and discriminative power, and large datasets, which makes exhaustive computations intractable.

In this paper we introduce a novel training scheme, based on mining of both positive and negative correspondences, and obtain large performance gains in patch retrieval. Our models generalize well across different datasets and applications, including wide-baseline matching, non-rigid deformations and extreme illumination changes. They can be used as drop-in replacement for traditional descriptors, e.g. SIFT, and are publicly available.

References

[1] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *ECCV*, 2006. 1, 2

[2] J. Bromley, I. Guyon, Y. Lecun, E. Sckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *NIPS*, 1994. 1, 3

[3] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *PAMI*, 33(1):43–57, 2011. 1, 2, 3, 4, 5, 7

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015. 2

[5] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A Matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 2

[6] E. Culurciello, J. Jin, A. Dundar, and J. Bates. An analysis of the connections between layers of deep neural networks. *CoRR*, abs/1306.0152, 2013. 3

[7] J. Davis and M. Goadrich. The relationship between PR and ROC curves. In *ICML*, 2006. 4

[8] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010. 2

[9] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A Procrustean approach to learning binary codes for large-scale image retrieval. In *PAMI*, 2012. 2

[10] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015. 2

[11] M. Jahrer, M. Grabner, and H. Bischof. Learned local descriptors for recognition and matching. In *Computer Vision Winter Workshop*, 2008. 2

[12] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 3

[13] I. Kokkinos, M. Bronstein, and A. Yuille. Dense scale-invariant descriptors for images and surfaces. In *INRIA Research Report 7914*, 2012. 1

[14] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004. 1, 2, 3, 4, 7, 8

[17] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005. 3

[18] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *ICML*, 2009. 3

[19] C. Osendorfer, J. Bayer, S. Urban, and P. van der Smagt. Convolutional neural networks learn compact local image descriptors. In *ICONIP*, volume 8228. 2013. 2

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *ICCV*, 2011. 1

[21] P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *ICPR*, 2012. 3

[22] E. Simo-Serra, C. Torras, and F. Moreno-Noguer. DaLI: Deformation and Light Invariant Descriptor. *IJCV*, 2015. 1, 2, 7, 8

[23] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*, 2014. 1, 2, 5, 6, 7, 8

[24] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. In *PAMI*, volume 34, 2012. 2

[25] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008. 7, 8

[26] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, 2013. 1, 2

[27] E. Tola, V. Lepetit, and P. Fua. DAISY: An efficient dense descriptor applied to wide baseline stereo. *PAMI*, 32(5):815–830, May 2010. 1, 2, 7, 8

[28] E. Trulls, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer. Dense segmentation-aware descriptors. *CVPR*, 2013. 1

[29] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting binary keypoint descriptors. In *CVPR*, 2013. 1, 2, 5, 6

[30] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. In *JMLR*, 2008. 1

[31] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org`, 2008. 6

[32] Z. Wang, B. Fan, and F. Wu. Local intensity order pattern for feature description. In *ICCV*, 2011. 1

[33] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 2

[34] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015. 2