LIFT: Learned Invariant Feature Transform

Kwang Moo Yi[1(✉)], Eduard Trulls[1], Vincent Lepetit[2], and Pascal Fua[1]

[1] Computer Vision Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{kwang.yi,eduard.trulls,pascal.fua}@epfl.ch
[2] Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria
lepetit@icg.tugraz.at

Abstract. We introduce a novel Deep Network architecture that implements the full feature point handling pipeline, that is, detection, orientation estimation, and feature description. While previous works have successfully tackled each one of these problems individually, we show how to learn to do all three in a unified manner while preserving end-to-end differentiability. We then demonstrate that our Deep pipeline outperforms state-of-the-art methods on a number of benchmark datasets, without the need of retraining.

Keywords: Local features · Feature descriptors · Deep Learning

1 Introduction

Local features play a key role in many Computer Vision applications. Finding and matching them across images has been the subject of vast amounts of research. Until recently, the best techniques relied on carefully hand-crafted features [1–5]. Over the past few years, as in many areas of Computer Vision, methods based in Machine Learning, and more specifically Deep Learning, have started to outperform these traditional methods [6–10].

These new algorithms, however, address only a single step in the complete processing chain, which includes detecting the features, computing their orientation, and extracting robust representations that allow us to match them across images. In this paper we introduce a novel Deep architecture that performs all three steps together. We demonstrate that it achieves better overall performance than the state-of-the-art methods, in large part because it allows these individual steps to be optimized to perform well in conjunction with each other.

K.M. Yi, E. Trulls—Equally contributed.

This work was supported in part by the EU FP7 project MAGELLAN under grant number ICT-FP7-611526.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-46466-4_28) contains supplementary material, which is available to authorized users.

B. Leibe et al. (Eds.): ECCV 2016, Part VI, LNCS 9910, pp. 467–483, 2016.
DOI: 10.1007/978-3-319-46466-4_28

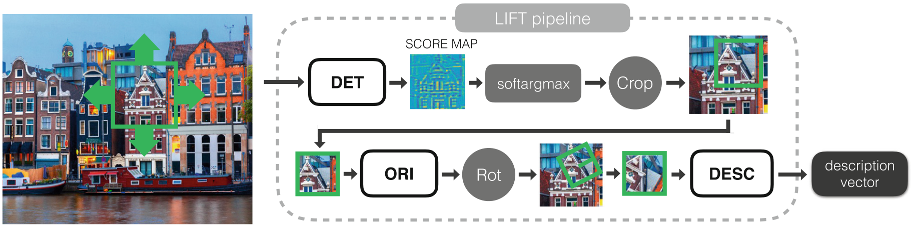

Fig. 1. Our integrated feature extraction pipeline. Our pipeline consists of three major components: the Detector, the Orientation Estimator, and the Descriptor. They are tied together with differentiable operations to preserve end-to-end differentiability. (Figures are best viewed in color.) (Color figure online)

Our architecture, which we refer to as LIFT for Learned Invariant Feature Transform, is depicted by Fig. 1. It consists of three components that feed into each other: the Detector, the Orientation Estimator, and the Descriptor. Each one is based on Convolutional Neural Networks (CNNs), and patterned after recent ones [6,9,10] that have been shown to perform these individual functions well. To mesh them together we use Spatial Transformers [11] to rectify the image patches given the output of the Detector and the Orientation Estimator. We also replace the traditional approaches to non-local maximum suppression (NMS) by the soft argmax function [12]. This allows us to preserve end-to-end differentiability, and results in a full network that can still be trained with backpropagation, which is not the case of any other architecture we know of.

Also, we show *how* to learn such a pipeline in an effective manner. To this end, we build a Siamese network and train it using the feature points produced by a Structure-from-Motion (SfM) algorithm that we ran on images of a scene captured under different viewpoints and lighting conditions, to learn its weights. We formulate this training problem on image patches extracted at different scales to make the optimization tractable. In practice, we found it impossible to train the full architecture from scratch, because the individual components try to optimize for different objectives. Instead, we introduce a problem-specific learning approach to overcome this problem. It involves training the Descriptor first, which is then used to train the Orientation Estimator, and finally the Detector, based on the already learned Descriptor and Orientation Estimator, differentiating through the entire network. At test time, we decouple the Detector, which runs over the whole image in scale space, from the Orientation Estimator and Descriptor, which process only the keypoints.

In the next section we briefly discuss earlier approaches. We then present our approach in detail and show that it outperforms many state-of-the-art methods.

2 Related Work

The amount of literature relating to local features is immense, but it always revolves about finding feature points, computing their orientation, and matching them. In this section, we will therefore discuss these three elements separately.

2.1 Feature Point Detectors

Research on feature point detection has focused mostly on finding distinctive locations whose scale and rotation can be reliably estimated. Early works [13, 14] used first-order approximations of the image signal to find corner points in images. FAST [15] used Machine Learning techniques but only to speed up the process of finding corners. Other than corner points, SIFT [1] detect blobs in scale-space; SURF [2] use Haar filters to speed up the process; Maximally Stable Extremal Regions (MSER) [16] detect regions; [17] detect affine regions. SFOP [18] use junctions and blobs, and Edge Foci [19] use edges for robustness to illumination changes. More recently, feature points based on more sophisticated and carefully designed filter responses [5,20] have also been proposed to further enhance the performance of feature point detectors.

In contrast to these approaches that focus on better engineering, and following the early attempts in learning detectors [21,22], [6] showed that a detector could be learned to deliver significantly better performance than the state-of-the-art. In this work, piecewise-linear convolutional filters are learned to robustly detect feature points in spite of lighting and seasonal changes. Unfortunately, this was done only for a single scale and from a dataset without viewpoint changes. We therefore took our inspiration from it but had to extend it substantially to incorporate it into our pipeline.

2.2 Orientation Estimation

Despite the fact that it plays a critical role in matching feature points, the problem of estimating a discriminative orientation has received noticeably less attention than detection or feature description. As a result, the method introduced by SIFT [1] remains the *de facto* standard up to small improvements, such as the fact that it can be sped-up by using the intensity centroid, as in ORB [4].

A departure from this can be found in a recent paper [9] that introduced a Deep Learning-based approach to predicting stable orientations. This resulted in significant gains over the state-of-the-art. We incorporate this architecture into our pipeline and show how to train it using our problem-specific training strategy, given our learned descriptors.

2.3 Feature Descriptors

Feature descriptors are designed to provide discriminative representations of salient image patches, while being robust to transformations such as viewpoint or illumination changes. The field reached maturity with the introduction of SIFT [1], which is computed from local histograms of gradient orientations, and SURF [2], which uses integral image representations to speed up the computation. Along similar lines, DAISY [3] relies on convolved maps of oriented gradients to approximate the histograms, which yields large computational gains when extracting dense descriptors.

Even though they have been extremely successful, these hand-crafted descriptors can now be outperformed by newer ones that have been learned. These range from unsupervised hashing to supervised learning techniques based on linear discriminant analysis [23,24], genetic algorithm [25], and convex optimization [26]. An even more recent trend is to extract features directly from raw image patches with CNNs trained on large volumes of data. For example, MatchNet [7] trained a Siamese CNN for feature representation, followed by a fully-connected network to learn the comparison metric. DeepCompare [8] showed that a network that focuses on the center of the image can increase performance. The approach of [27] relied on a similar architecture to obtain state-of-the-art results for narrow-baseline stereo. In [10], hard negative mining was used to learn compact descriptors that use on the Euclidean distance to measure similarity. The algorithm of [28] relied on sample triplets to mine hard negatives.

In this work, we rely on the architecture of [10] because the corresponding descriptors are trained and compared with the Euclidean distance, which has a wider range of applicability than descriptors that require a learned metric.

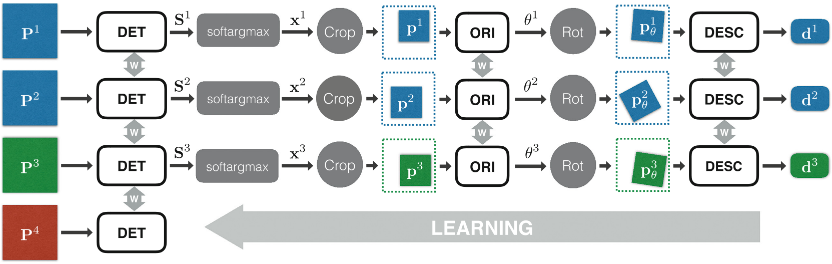

Fig. 2. Our Siamese training architecture with four branches, which takes as input a quadruplet of patches: Patches $\mathbf{P}^1$ and $\mathbf{P}^2$ (blue) correspond to different views of the same physical point, and are used as positive examples to train the Descriptor; $\mathbf{P}^3$ (green) shows a different 3D point, which serves as a negative example for the Descriptor; and $\mathbf{P}^4$ (red) contains no distinctive feature points and is only used as a negative example to train the Detector. Given a patch $\mathbf{P}$, the Detector, the softargmax, and the Spatial Transformer layer Crop provide all together a smaller patch $\mathbf{p}$ inside $\mathbf{P}$. $\mathbf{p}$ is then fed to the Orientation Estimator, which along with the Spatial Transformer layer Rot, provides the rotated patch $\mathbf{p}_\theta$ that is processed by the Descriptor to obtain the final description vector $\mathbf{d}$. (Color figure online)

3 Method

In this section, we first formulate the entire feature detection and description pipeline in terms of the Siamese architecture depicted by Fig. 2. Next, we discuss the type of data we need to train our networks and how to collect it. We then describe the training procedure in detail.

3.1 Problem Formulation

We use image patches as input, rather than full images. This makes the learning scalable without loss of information, as most image regions do not contain keypoints. The patches are extracted from the keypoints used by a SfM pipeline, as will be discussed in Sect. 3.2. We take them to be small enough that we can assume they contain only one dominant local feature at the given scale, which reduces the learning process to finding the most distinctive point in the patch.

To train our network we create the four-branch Siamese architecture pictured in Fig. 2. Each branch contains three distinct CNNs, a Detector, an Orientation Estimator, and a Descriptor. For training purposes, we use quadruplets of image patches. Each one includes two image patches $\mathbf{P}^1$ and $\mathbf{P}^2$, that correspond to different views of the same 3D point, one image patch $\mathbf{P}^3$, that contains the projection of a different 3D point, and one image patch $\mathbf{P}^4$ that does not contain any distinctive feature point. During training, the i-th patch $\mathbf{P}^i$ of each quadruplet will go through the i-th branch.

To achieve end-to-end differentiability, the components of each branch are connected as follows:

1. Given an input image patch $\mathbf{P}$, the Detector provides a score map $\mathbf{S}$.
2. We perform a soft argmax [12] on the score map $\mathbf{S}$ and return the location $\mathbf{x}$ of a single potential feature point.
3. We extract a smaller patch $\mathbf{p}$ centered on $\mathbf{x}$ with the Spatial Transformer layer Crop (Fig. 2). This serves as the input to the Orientation Estimator.
4. The Orientation Estimator predicts a patch orientation θ.
5. We rotate $\mathbf{p}$ according to this orientation using a second Spatial Transformer layer, labeled as Rot in Fig. 2, to produce $\mathbf{p}_\theta$.
6. $\mathbf{p}_\theta$ is fed to the Descriptor network, which computes a feature vector $\mathbf{d}$.

Note that the Spatial Transformer layers are used only to manipulate the image patches while preserving differentiability. They are not learned modules. Also, both the location $\mathbf{x}$ proposed by the Detector and the orientation θ for the patch proposal are treated implicitly, meaning that we let the entire network discover distinctive locations and stable orientations while learning.

Since our network consists of components with different purposes, learning the weights is non-trivial. Our early attempts at training the network as a whole from scratch were unsuccessful. We therefore designed a problem-specific learning approach that involves learning first the Descriptor, then the Orientation Estimator given the learned descriptor, and finally the Detector, conditioned on the other two. This allows us to tune the Orientation Estimator for the Descriptor, and the Detector for the other two components.

We will elaborate on this learning strategy in Sects. 3.3 (Descriptor), 3.4 (Orientation Estimator), and 3.5 (Detector), that is, in the order they are learned.

3.2 Creating the Training Dataset

There are datasets that can be used to train feature descriptors [24] and orientation estimators [9]. However it is not so clear how to train a keypoint detector, and the vast majority of techniques still rely on hand-crafted features. The TILDE detector [6] is an exception, but the training dataset does not exhibit any viewpoint changes.

To achieve invariance we need images that capture views of the same scene under different illumination conditions and seen from different perspectives. We thus turned to photo-tourism image sets. We used the collections from Piccadilly Circus in London and the Roman Forum in Rome from [29] to reconstruct the 3D using VisualSFM [30], which relies of SIFT features. *Piccadilly* contains 3384 images, and the reconstruction has 59k unique points with an average of 6.5 observations for each. *Roman-Forum* contains 1658 images and 51k unique points, with an average of 5.2 observations for each. Figure 3 shows some examples.

We split the data into training and validation sets, discarding views of training points on the validation set and vice-versa. To build the positive training samples we consider only the feature points that survive the SfM reconstruction process. To extract patches that do not contain any distinctive feature point, as required by our training method, we randomly sample image regions that contain no SIFT features, including those that were not used by SfM.

We extract grayscale training patches according to the scale σ of the point, for both feature and non-feature point image regions. Patches $\mathbf{P}$ are extracted from a $24\sigma \times 24\sigma$ support region at these locations, and standardized into $S \times S$ pixels where $S = 128$. The smaller patches $\mathbf{p}$ and $\mathbf{p}_\theta$ that serve as input to the Orientation Estimator and the Descriptor, are cropped and rotated versions of these patches, each having size $s \times s$, where $s = 64$. The smaller patches effectively

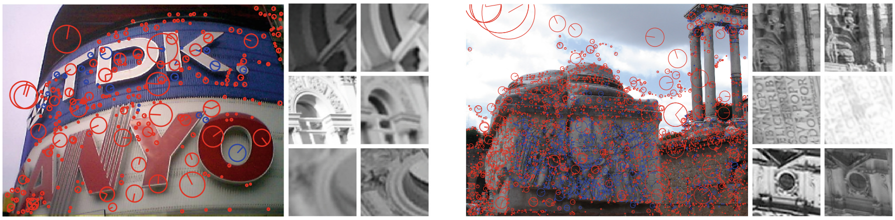

Fig. 3. Sample images and patches from *Piccadilly* (left) and *Roman-Forum* (right). Keypoints that survive the SfM pipeline are drawn in blue, and the rest in red. (Color figure online)

correspond to the SIFT descriptor support region size of 12σ. To avoid biasing the data, we apply uniform random perturbations to the patch location with a range of 20 % (4.8σ). Finally, we normalize the patches with the grayscale mean and standard deviation of the entire training set.

3.3 Descriptor

Learning feature descriptors from raw image patches has been extensively researched during the past year [7,8,10,27,28,31], with multiple works reporting impressive results on patch retrieval, narrow baseline stereo, and matching non-rigid deformations. Here we rely on the relatively simple networks of [10], with three convolutional layers followed by hyperbolic tangent units, l_2 pooling [32] and local subtractive normalization, as they do not require learning a metric.

The Descriptor can be formalized simply as

$$\mathbf{d} = h_\rho(\mathbf{p}_\theta) \,, \tag{1}$$

where $h(.)$ denotes the Descriptor CNN, ρ its parameters, and $\mathbf{p}_\theta$ is the rotated patch from the Orientation Estimator. When training the Descriptor, we do not yet have the Detector and the Orientation Estimator trained. We therefore use the image locations and orientations of the feature points used by the SfM to generate image patches $\mathbf{p}_\theta$.

We train the Descriptor by minimizing the sum of the loss for pairs of corresponding patches $\left(\mathbf{p}_\theta^1, \mathbf{p}_\theta^2\right)$ and the loss for pairs of non-corresponding patches $\left(\mathbf{p}_\theta^1, \mathbf{p}_\theta^3\right)$. The loss for pair $\left(\mathbf{p}_\theta^k, \mathbf{p}_\theta^l\right)$ is defined as the hinge embedding loss of the Euclidean distance between their description vectors. We write

$$\mathcal{L}_{\text{desc}}(\mathbf{p}_\theta^k, \mathbf{p}_\theta^l) = \begin{cases} \left\| h_\rho\left(\mathbf{p}_\theta^k\right) - h_\rho\left(\mathbf{p}_\theta^l\right) \right\|_2 & \text{for positive pairs, and} \\ \max\left(0, C - \left\| h_\rho\left(\mathbf{p}_\theta^k\right) - h_\rho\left(\mathbf{p}_\theta^l\right) \right\|_2\right) & \text{for negative pairs} \,, \end{cases} \tag{2}$$

where positive and negative samples are pairs of patches that do or do not correspond to the same physical 3D points, $\|\cdot\|_2$ is the Euclidean distance, and $C = 4$ is the margin for embedding.

We use hard mining during training, which was shown in [10] to be critical for descriptor performance. Following this methodology, we forward K_f sample pairs and use only the K_b pairs with the highest training loss for back-propagation, where $r = K_f / K_b \geq 1$ is the 'mining ratio'. In [10] the network was pre-trained without mining and then fine-tuned with $r = 8$. Here, we use an increasing mining scheme where we start with $r = 1$ and double the mining ratio every 5000 batches. We use balanced batches with 128 positive pairs and 128 negative pairs, mining each separately.

3.4 Orientation Estimator

Our Orientation Estimator is inspired by that of [9]. However, this specific one requires pre-computations of description vectors for multiple orientations to compute numerically the Jacobian of the method parameters with respect to orientations. This is a critical limitation for us because we treat the output of the detector component implicitly throughout the pipeline and it is thus not possible to pre-compute the description vectors.

We therefore propose to use Spatial Transformers [11] instead to learn the orientations. Given a patch $\mathbf{p}$ from the region proposed by the detector, the Orientation Estimator predicts an orientation

$$\theta = g_\phi(\mathbf{p}) \;, \tag{3}$$

where g denotes the Orientation Estimator CNN, and ϕ its parameters.

Together with the location $\mathbf{x}$ from the Detector and $\mathbf{P}$ the original image patch, θ is then used by the second Spatial Transformer Layer $\mathrm{Rot}(.)$ to provide a patch $\mathbf{p}_\theta = \mathrm{Rot}\,(\mathbf{P}, \mathbf{x}, \theta)$, which is the rotated version of patch $\mathbf{p}$.

We train the Orientation Estimator to provide the orientations that minimize the distances between description vectors for different views of the same 3D points. We use the already trained Descriptor to compute the description vectors, and as the Detector is still not trained, we use the image locations from SfM. More formally, we minimize the loss for pairs of corresponding patches, defined as the Euclidean distance between their description vectors

$$\mathcal{L}_{\text{orientation}}(\mathbf{P}^1, \mathbf{x}^1, \mathbf{P}^2, \mathbf{x}^2) = \left\| h_\rho(G(\mathbf{P}^1, \mathbf{x}^1)) - h_\rho(G(\mathbf{P}^2, \mathbf{x}^2)) \right\|_2 \;, \tag{4}$$

where $G(\mathbf{P}, \mathbf{x})$ is the patch centered on $\mathbf{x}$ after orientation correction: $G(\mathbf{P}, \mathbf{x}) = \mathrm{Rot}(\mathbf{P}, \mathbf{x}, g_\phi(\mathrm{Crop}(\mathbf{P}, \mathbf{x})))$. This complex notation is necessary to properly handle the cropping of the image patches. Recall that pairs $(\mathbf{P}^1, \mathbf{P}^2)$ comprise image patches containing the projections of the same 3D point, and locations $\mathbf{x}^1$ and $\mathbf{x}^2$ denote the reprojections of these 3D points. As in [9], we do not use pairs that correspond to different physical points whose orientations are not related.

3.5 Detector

The Detector takes an image patch as input, and returns a score map. We implement it as a convolution layer followed by piecewise linear activation functions, as in TILDE [6]. More precisely, the score map $\mathbf{S}$ for patch $\mathbf{P}$ is computed as:

$$\mathbf{S} = f_\mu(\mathbf{P}) = \sum_n^N \delta_n \max_m^M (\mathbf{W}_{mn} * \mathbf{P} + \mathbf{b}_{mn}) \;, \tag{5}$$

where $f_\mu(\mathbf{P})$ denotes the Detector itself with parameters μ, δ_n is $+1$ if n is odd and -1 otherwise, μ is made of the filters W_{mn} and biases b_{mn} of the convolution layer to learn, $*$ denotes the convolution operation, and N and M are hyper-parameters controlling the complexity of the piecewise linear activation function.

The main difference with TILDE lies in the way we train this layer. To let $\mathbf{S}$ have maxima in places other than a fixed location retrieved by SfM, we treat this location implicitly, as a latent variable. Our method can potentially discover points that are more reliable and easier to learn, whereas [6] cannot. Incidentally, in our early experiments, we noticed that it was harmful to force the Detector to optimize directly for SfM locations.

From the score map $\mathbf{S}$, we obtain the location $\mathbf{x}$ of a feature point as

$$\mathbf{x} = \text{softargmax}\left(\mathbf{S}\right) \ , \tag{6}$$

where softargmax is a function which computes the Center of Mass with the weights being the output of a standard softmax function [12]. We write

$$\text{softargmax}\left(\mathbf{S}\right) = \frac{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))\mathbf{y}}{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))} \ , \tag{7}$$

where $\mathbf{y}$ are locations in $\mathbf{S}$, and $\beta = 10$ is a hyper-parameter controlling the smoothness of the softargmax. This softargmax function acts as a differentiable version of non-maximum suppression. $\mathbf{x}$ is given to the first Spatial Transformer Layer Crop(.) together with the patch $\mathbf{P}$ to extract a smaller patch $\mathbf{p} = \text{Crop}\left(\mathbf{P}, \mathbf{x}\right)$ used as input to the Orientation Estimator.

As the Orientation Estimator and the Descriptor have been learned by this point, we can train the Detector given the full pipeline. To optimize over the parameters μ, we minimize the distances between description vectors for the pairs of patches that correspond to the same physical points, while maximizing the classification score for patches not corresponding to the same physical points.

More exactly, given training quadruplets $\left(\mathbf{P}^1, \mathbf{P}^2, \mathbf{P}^3, \mathbf{P}^4\right)$, where $\mathbf{P}^1$ and $\mathbf{P}^2$ correspond to the same physical point, $\mathbf{P}^1$ and $\mathbf{P}^3$ correspond to different SfM points, and $\mathbf{P}^4$ to a non-feature point location, we minimize the sum of their loss functions

$$\mathcal{L}_{\text{detector}}(\mathbf{P}^1, \mathbf{P}^2, \mathbf{P}^3, \mathbf{P}^4) = \gamma \mathcal{L}_{class}(\mathbf{P}^1, \mathbf{P}^2, \mathbf{P}^3, \mathbf{P}^4) + \mathcal{L}_{pair}(\mathbf{P}^1, \mathbf{P}^2) \ , \tag{8}$$

where γ is a hyper-parameter balancing the two terms in this summation

$$\mathcal{L}_{\text{class}}(\mathbf{P}^1, \mathbf{P}^2, \mathbf{P}^3, \mathbf{P}^4) = \sum_{i=1}^{4} \alpha_i \max\left(0, \left(1 - \text{softmax}\left(f_\mu\left(\mathbf{P}^i\right)\right) y_i\right)\right)^2 \ , \tag{9}$$

with $y_i = -1$ and $\alpha_i = 3/6$ if $i = 4$, and $y_i = +1$ and $\alpha_i = 1/6$ otherwise to balance the positives and negatives. softmax is the log-mean-exponential softmax function. We write

$$\begin{aligned} \mathcal{L}_{\text{pair}}(\mathbf{P}^1, \mathbf{P}^2) = \| \ & h_\rho(G(\mathbf{P}^1, \text{softargmax}(f_\mu(\mathbf{P}^1)))) \\ - \ & h_\rho(G(\mathbf{P}^2, \text{softargmax}(f_\mu(\mathbf{P}^2)))) \ \|_2 \ . \end{aligned} \tag{10}$$

Note that the locations of the detected feature points $\mathbf{x}$ appear only implicitly and are discovered during training. Furthermore, all three components are tied in with the Detector learning. As with the Descriptor we use a hard mining strategy, in this case with a fixed mining ratio of $r = 4$.

In practice, as the Descriptor already learns some invariance, it can be hard for the Detector to find new points to learn implicitly. To let the Detector start with an idea of the regions it should find, we first constrain the patch proposals $\mathbf{p} = \text{Crop}(\mathbf{P}, \text{softargmax}(f_\mu(\mathbf{P})))$ that correspond to the same physical points to overlap. We then continue training the Detector without this constraint.

Specifically, when pre-training the Detector, we replace $\mathcal{L}_{\text{pair}}$ in Eq. (8) with $\tilde{\mathcal{L}}_{\text{pair}}$, where $\tilde{\mathcal{L}}_{\text{pair}}$ is equal to 0 when the patch proposals overlap exactly, and increases with the distance between them otherwise. We therefore write

$$\tilde{\mathcal{L}}_{\text{pair}}(\mathbf{P}^1, \mathbf{P}^2) = 1 - \frac{\mathbf{p}^1 \cap \mathbf{p}^2}{\mathbf{p}^1 \cup \mathbf{p}^2} + \frac{\max\left(0, \left\|\mathbf{x}^1 - \mathbf{x}^2\right\|_1 - 2s\right)}{\sqrt{\mathbf{p}^1 \cup \mathbf{p}^2}}\,, \tag{11}$$

where $\mathbf{x}^j = \text{softargmax}(f_\mu(\mathbf{P}^j))$, $\mathbf{p}^j = \text{Crop}(\mathbf{P}^j, \mathbf{x}^j)$, $\|\cdot\|_1$ is the l_1 norm. Recall that $s = 64$ pixels is the width and height of the patch proposals.

3.6 Runtime Pipeline

The pipeline used at run-time is shown in Fig. 4. As our method is trained on patches, simply applying it over the image would require the network to be tested with a sliding window scheme over the whole image. In practice, this would be too expensive. Fortunately, as the Orientation Estimator and the Descriptor only need to be run at local maxima, we can simply decouple the detector from the rest to apply it to the full image, and replace the softargmax function by NMS, as outlined in red in Fig. 4. We then apply the Orientation Estimator and the Descriptor only to the patches centered on local maxima.

More exactly, we apply the Detector independently to the image at different resolutions to obtain score maps in scale space. We then apply a traditional NMS scheme similar to that of [1] to detect feature point locations.

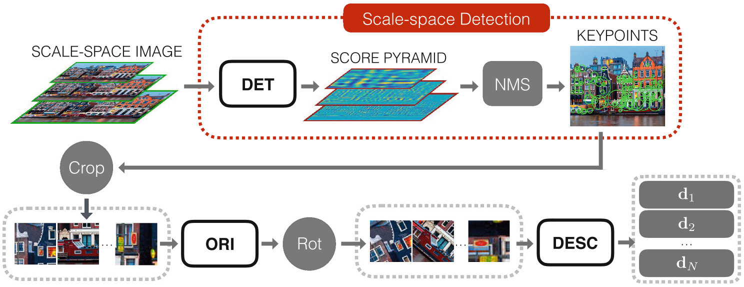

Fig. 4. An overview of our runtime architecture. As the Orientation Estimator and the Descriptor only require evaluation at local maxima, we decouple the Detector and run it in scale space with traditional NMS to obtain proposals for the two other components. (Color figure online)

4 Experimental Validation

In this section, we first present the datasets and metrics we used. We then present qualitative results, followed by a thorough quantitative comparison against a number of state-of-the-art baselines, which we consistently outperform.

Finally, to better understand what elements of our approach most contribute to this result, we study the importance of the pre-training of the Detector component, discussed in Sect. 3.5, and analyze the performance gains attributable to each component.

4.1 Dataset and Experimental Setup

We evaluate our pipeline on three standard datasets:

- The *Strecha* dataset [33], which contains 19 images of two scenes seen from increasingly different viewpoints.
- The *DTU* dataset [34], which contains 60 sequences of objects with different viewpoints and illumination settings. We use this dataset to evaluate our method under viewpoint changes.
- The *Webcam* dataset [6], which contains 710 images of 6 scenes with strong illumination changes but seen from the same viewpoint. We use this dataset to evaluate our method under natural illumination changes.

For *Strecha* and *DTU* we use the provided ground truth to establish correspondences across viewpoints. We use a maximum of 1000 keypoints per image, and follow the standard evaluation protocol of [35] on the common viewpoint region. This lets us evaluate the following metrics.

- Repeatability (Rep.): Repeatability of feature points, expressed as a ratio. This metric captures the performance of the feature point detector by reporting the ratio of keypoints that are found consistently in the shared region.
- Nearest Neighbor mean Average Precision (NN mAP): Area Under Curve (AUC) of the Precision-Recall curve, using the Nearest Neighbor matching strategy. This metric captures how discriminating the descriptor is by evaluating it at multiple descriptor distance thresholds.
- Matching Score (M. Score): The ratio of ground truth correspondences that can be recovered by the whole pipeline over the number of features proposed by the pipeline in the shared viewpoint region. This metric measures the overall performance of the pipeline.

We compare our method on the three datasets to the following combination of feature point detectors and descriptors, as reported by the authors of the corresponding papers: SIFT [1], SURF [2], KAZE [36], ORB [4], Daisy [37] with SIFT detector, sGLOH [38] with Harris-affine detector [39], MROGH [40] with Harris-affine detector, LIOP [41] with Harris-affine detector, BiCE [42] with Edge Foci detector [19], BRISK [43], FREAK [44] with BRISK detector, VGG [26] with SIFT detector, DeepDesc [10] with SIFT detector, PN-Net [28] with SIFT detector, and MatchNet [7] with SIFT detector. We also consider SIFT with Hessian-Affine keypoints [17]. For the learned descriptors VGG, DeepDesc, PN-Net and MatchNet we use SIFT keypoints because they are trained using a dataset created with Difference-of-Gaussians, which is essentially the same as SIFT. In the case of Daisy, which was not developed for a specific detector, we also use SIFT

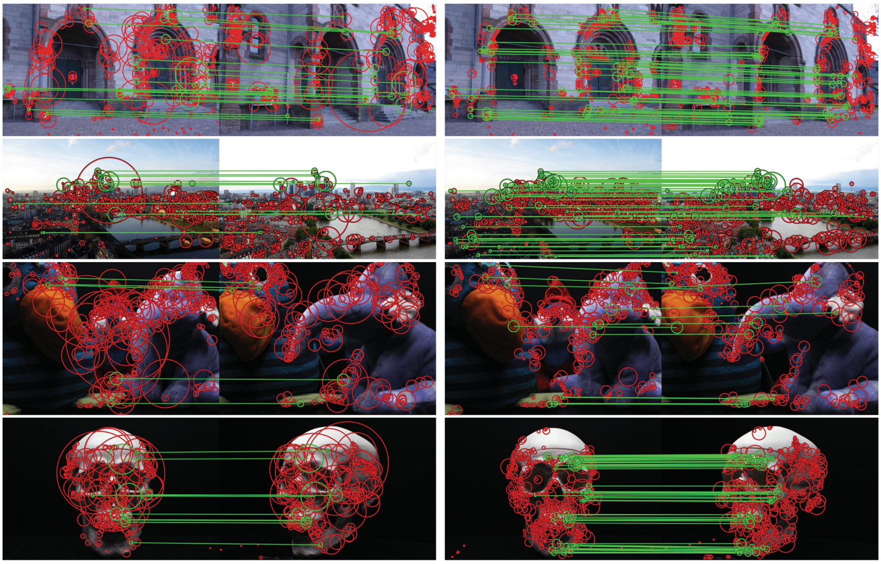

Fig. 5. Qualitative local feature matching examples of **left:** SIFT and **right:** our method LIFT. Correct matches recovered by each method are shown in green lines and the descriptor support regions with red circles. **Top row:** *Herz-Jesu-P8* of *Strecha*, **second row:** *Frankfurt* of *Webcam*, **third row:** *Scene 7* of *DTU* and **bottom row:** *Scene 19* of *DTU*. Note that the images are very different from one another. (Color figure online)

keypoints. To make our results reproducible, we provide additional implementation details for LIFT and the baselines in the supplementary material.[1]

4.2 Qualitative Examples

Figure 5 shows image matching results with 500 feature points, for both SIFT and our LIFT pipeline trained with *Piccadilly*. As expected, LIFT returns more correct correspondences across the two images. One thing to note is that the two DTU scenes in the bottom two rows are completely different from the phototourism datasets we used for training. Given that the two datasets are very different, this shows good generalization properties.

4.3 Quantitative Evaluation of the Full Pipeline

Figure 6 shows the average matching score for all three datasets, and Table 1 provides the exact numbers for the two LIFT variants. LIFT (pic) is trained with *Piccadilly* and LIFT (rf) with *Roman-Forum*. Both of our learned models significantly outperform the state-of-the-art on *Strecha* and *DTU* and achieve

[1] Source and models will be available at https://github.com/cvlab-epfl/LIFT.

Table 1. Average matching score for all baselines.

	SIFT	SIFT-HesAff	SURF	ORB	Daisy	sGLOH	MROGH	LIOP	BiCE
Strecha	.283	.314	.208	.157	.272	.207	.239	.211	.270
DTU	.272	.274	.244	.127	.262	.187	.223	.189	.242
Webcam	.128	.164	.117	.120	.120	.113	.125	.086	.166
	BRISK	FREAK	VGG	MatchNet	DeepDesc	PN-Net	KAZE	LIFT-pic	LIFT-rf
Strecha	.208	.183	.300	.223	.298	.300	.250	**.374**	.369
DTU	.193	.186	.271	.198	.257	.267	.213	**.317**	.308
Webcam	.118	.116	.118	.101	.116	.114	.195	.196	**.202**

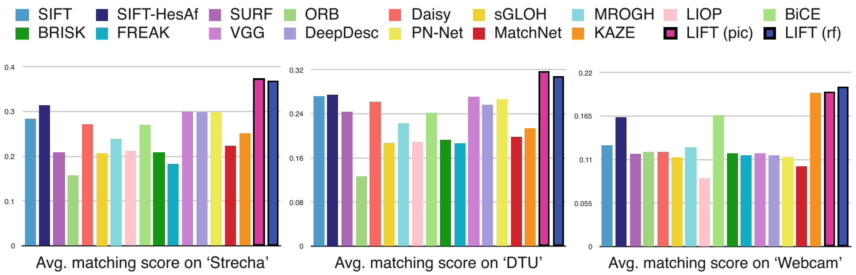

Fig. 6. Average matching score for all baselines.

state-of-the-art on *Webcam*. Note that KAZE, which is the best performing competitor on *Webcam*, performs poorly on the other two datasets. As discussed above, *Piccadilly* and *Roman-Forum* are very different from the datasets used for testing. This underlines the strong generalization capability of our approach, which is not always in evidence with learning-based methods.

Interestingly, on *DTU*, SIFT is still the best performing method among the competitors, even compared to methods that rely on Deep Learning, such as DeepDesc and PN-Net. Also, the gap between SIFT and the learning-based VGG, DeepDesc, and PN-Net is not large for the *Strecha* dataset.

These results show that although a component may outperform another method when evaluated individually, they may fail to deliver their full potential when integrated into the full pipeline, which is what really matters. In other words, it is important to learn the components together, as we do, and to consider the whole pipeline when evaluating feature point detectors and descriptors.

Table 2. Results on *Strecha* for both LIFT models trained on *Piccadilly* and *Roman-Forum*, with the pre-trained and fully-trained Detector.

	Trained on *Piccadilly*		Trained on *Roman-Forum*	
	Rep	M.Score	Rep	M.Score
Pre-trained	.436	.367	.447	.368
Fully-trained	**.446**	**.374**	**.447**	**.369**

Table 3. Results on *Strecha* for both LIFT models trained on *Piccadilly* and *Roman-Forum*, interchanging our components with their SIFT counterparts.

			Trained on *Piccadilly*			Trained on *Roman-Forum*		
Det.	Ori.	Desc.	Rep.	NN mAP	M.Score	Rep.	NN mAP	M.Score
SIFT	SIFT	SIFT	.428	.517	.282	.428	.517	.282
SIFT	Ours	SIFT		.671	.341		.662	.338
SIFT	SIFT	Ours		.568	.290		.581	.295
SIFT	Ours	Ours		.685	.344		**.688**	.342
Ours	SIFT	SIFT	**.446**	.540	.325	**.447**	.545	.319
Ours	Ours	SIFT		.644	.372		.630	.360
Ours	SIFT	Ours		.629	.339		.644	.337
Ours	Ours	Ours	**.446**	**.686**	**.374**	**.447**	.683	**.369**

4.4 Performance of Individual Components

Fine-Tuning the Detector. Recall that we pre-train the detector and then finalize the training with the Orientation Estimator and the Descriptor, as discussed in Sect. 3.5. It is therefore interesting to see the effect of this finalizing stage. In Table 2 we evaluate the entire pipeline with the pre-trained Detector and the final Detector. As the pair-wise loss term $\tilde{\mathcal{L}}_{\text{pair}}$ of Eq. (11) is designed to emulate the behavior of an ideal descriptor, the pre-trained Detector already performs well. However, the full training pushes the performance slightly higher.

A closer look at Table 2 reveals that gains are larger overall for *Piccadilly* than for *Roman-Forum*. This is probably due to the fact that *Roman-Forum* does not have many non-feature point regions. In fact, the network started to over-fit quickly after a few iterations on this dataset. The same happened when we further tried to fine-tune the full pipeline as a whole, suggesting that our learning strategy is already providing a good global solution.

Performance of Individual Components. To understand the influence of each component on the overall performance, we exchange them with their SIFT counterparts, for both LIFT (pic) and LIFT (rf), on *Strecha*. We report the results in Table 3. In short, each time we exchange to SIFT, we decrease performance, thus showing that each element of the pipeline plays and important role. Our Detector gives higher repeatability for both models. Having better orientations also helps whichever detector or descriptor is being used, and also the Deep Descriptors perform better than SIFT.

One thing to note is that our Detector is not only better in terms of repeatability, but generally better in terms of both the NN mAP, which captures the descriptor performance, and in terms of matching score, which evaluates the full pipeline. This shows that our Detector learns to find not only points that can be found often but also points that can be matched easily, indicating that training the pipeline as a whole is important for optimal performance.

5 Conclusion

We have introduced a novel Deep Network architecture that combines the three components of standard pipelines for local feature detection and description into a single differentiable network. We used Spatial Transformers together with the softargmax function to mesh them together into a unified network that can be trained end-to-end with back-propagation. While this makes learning the network from scratch theoretically possible, it is not *practical*. We therefore proposed an effective strategy to train it.

Our experimental results demonstrate that our integrated approach outperforms the state-of-the-art. To further improve performance, we will look into strategies that allow us to take advantage even more effectively of our ability to train the network as a whole. In particular, we will look into using hard negative mining strategies over the whole image [45] instead of relying on pre-extracted patches. This has the potential of producing more discriminative filters and, consequently, better descriptors.

References

1. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **20**(2), 91–110 (2004)
2. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. CVIU **10**(3), 346–359 (2008)
3. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: CVPR (2008)
4. Rublee, E., Rabaud, V., Konolidge, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: ICCV (2011)
5. Mainali, P., Lafruit, G., Tack, K., Van Gool, L., Lauwereins, R.: Derivative-based scale invariant image feature detector with error resilience. TIP **23**(5), 2380–2391 (2014)
6. Verdie, Y., Yi, K.M., Fua, P., Lepetit, V.: TILDE: a temporally invariant learned DEtector. In: CVPR (2015)
7. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: MatchNet: unifying feature and metric learning for patch-based matching. In: CVPR (2015)
8. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: CVPR (2015)
9. Yi, K., Verdie, Y., Lepetit, V., Fua, P.: Learning to assign orientations to feature points. In: CVPR (2016)
10. Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: ICCV (2015)
11. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: NIPS (2015)
12. Chapelle, O., Wu, M.: Gradient descent optimization of smoothed information retrieval metrics. Inf. Retrieval **13**(3), 216–235 (2009)
13. Harris, C., Stephens, M.: A combined corner and edge detector. In: Fourth Alvey Vision Conference (1988)

14. Moravec, H.: Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University, Stanford University, September 1980
15. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006). doi:10.1007/11744023_34
16. Matas, J., Chum, O., Martin, U., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: BMVC, pp. 384–393, September 2002
17. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 128–142. Springer, Heidelberg (2002). doi:10.1007/3-540-47969-4_9
18. Förstner, W., Dickscheid, T., Schindler, F.: Detecting interpretable and accurate scale-invariant keypoints. In: ICCV, September 2009
19. Zitnick, C., Ramnath, K.: Edge foci interest points. In: ICCV (2011)
20. Mainali, P., Lafruit, G., Yang, Q., Geelen, B., Van Gool, L., Lauwereins, R.: SIFER: scale-invariant feature detector with error resilience. IJCV **104**(2), 172–197 (2013)
21. Šochman, J., Matas, J.: Learning a fast emulator of a binary decision process. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007. LNCS, vol. 4844, pp. 236–245. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76390-1_24
22. Trujillo, L., Olague, G.: Using evolution to learn how to perform interest point detection. In: ICPR, pp. 211–214 (2006)
23. Strecha, C., Bronstein, A., Bronstein, M., Fua, P.: LDAHash: improved matching with smaller descriptors. PAMI **34**(1), 66–78 (2012)
24. Winder, S., Brown, M.: Learning local image descriptors. In: CVPR, June 2007
25. Perez, C., Olague, G.: Genetic programming as strategy for learning image descriptor operators. Intell. Data Anal. **17**, 561–583 (2013)
26. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. PAMI **36**, 1573–1585 (2014)
27. Zbontar, J., LeCun, Y.: Computing the stereo matching cost with a convolutional neural network. In: CVPR (2015)
28. Balntas, V., Johns, E., Tang, L., Mikolajczyk, K.: PN-Net: conjoined triple deep network for learning local image descriptors. In: arXiv Preprint (2016)
29. Wilson, K., Snavely, N.: Robust global translations with 1DSfM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 61–75. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10578-9_5
30. Wu, C.: Towards linear-time incremental structure from motion. In: 3DV (2013)
31. Paulin, M., Douze, M., Harchaoui, Z., Mairal, J., Perronnin, F., Schmid, C.: Local convolutional features with unsupervised training for image retrieval. In: ICCV (2015)
32. Sermanet, P., Chintala, S., LeCun, Y.: Convolutional neural networks applied to house numbers digit classification. In: ICPR (2012)
33. Strecha, C., Hansen, W., Van Gool, L., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: CVPR (2008)
34. Aanaes, H., Dahl, A.L., Pedersen, K.S.: Interesting interest points. IJCV **97**, 18–35 (2012)
35. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. In: CVPR, pp. 257–263, June 2003
36. Alcantarilla, P.F., Bartoli, A., Davison, A.J.: KAZE features. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7577, pp. 214–227. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33783-3_16

37. Tola, E., Lepetit, V., Fua, P.: Daisy: an efficient dense descriptor applied to wide baseline stereo. PAMI **32**(5), 815–830 (2010)
38. Bellavia, F., Tegolo, D.: Improving sift-based descriptors stability to rotations. In: ICPR (2010)
39. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. IJCV **60**, 63–86 (2004)
40. Fan, B., Wu, F., Hu, Z.: Aggregating gradient distributions into intensity orders: a novel local image descriptor. In: CVPR (2011)
41. Wang, Z., Fan, B., Wu, F.: Local intensity order pattern for feature description. In: ICCV (2011)
42. Zitnick, C.L.: Binary coherent edge descriptors. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6312, pp. 170–182. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15552-9_13
43. Leutenegger, S., Chli, M., Siegwart, R.: BRISK: binary robust invariant scalable keypoints. In: ICCV (2011)
44. Alahi, A., Ortiz, R., Vandergheynst, P.: FREAK: fast retina keypoint. In: CVPR (2012)
45. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI **32**(9), 1627–1645 (2010)