Reinforced Feature Points:
Optimizing Feature Detection and Description for a High-Level Task

Aritra Bhowmik¹, Stefan Gumhold¹, Carsten Rother², Eric Brachmann²
¹ TU Dresden, ² Heidelberg University

Abstract

We address a core problem of computer vision: Detection and description of 2D feature points for image matching. For a long time, hand-crafted designs, like the seminal SIFT algorithm, were unsurpassed in accuracy and efficiency. Recently, learned feature detectors emerged that implement detection and description using neural networks. Training these networks usually resorts to optimizing low-level matching scores, often pre-defining sets of image patches which should or should not match, or which should or should not contain key points. Unfortunately, increased accuracy for these low-level matching scores does not necessarily translate to better performance in high-level vision tasks. We propose a new training methodology which embeds the feature detector in a complete vision pipeline, and where the learnable parameters are trained in an end-to-end fashion. We overcome the discrete nature of key point selection and descriptor matching using principles from reinforcement learning. As an example, we address the task of relative pose estimation between a pair of images. We demonstrate that the accuracy of a state-of-the-art learning-based feature detector can be increased when trained for the task it is supposed to solve at test time. Our training methodology poses little restrictions on the task to learn, and works for any architecture which predicts key point heat maps, and descriptors for key point locations.

1. Introduction

Finding and matching sparse 2D feature points across images has been a long-standing problem in computer vision [19]. Feature detection algorithms enable the creation of vivid 3D models from image collections [21, 56, 45], building maps for robotic agents [31, 32], recognizing places [44, 24, 35] and precise locations [25, 52, 41] as well as recognizing objects [26, 34, 38, 1, 2]. Naturally, the design of feature detection and description algorithms, subsumed as feature detection in the following, has received tremendous attention in computer vision research since its early days. Although invented three decades ago, the seminal SIFT algorithm [26] remains the gold standard feature detection pipeline to this day.
level matching benchmarks like H-Patches [3], such methods regularly achieve highly superior scores compared to a SIFT baseline. H-Patches [3] defines sets of matching image patches that undergo severe illumination and viewpoint changes. However, the increased accuracy in such matching tasks does not necessarily translate to increased accuracy in high-level vision pipelines. For example, we show that the state-of-the-art learned SuperPoint detector [14], while highly superior to SIFT [26] on H-Patches [3], does not reach SIFT’s capabilities when estimating an essential matrix for an image pair. Similar observations were reported in earlier studies, where the supposedly superior learned LIFT detector [58] failed to produce richer reconstructions than SIFT [26] in a structure-from-motion pipeline [46].

Some authors took notice of the discrepancy between low-level training and high-level performance, and developed training protocols that mimic properties of high-level vision pipelines. Lua et al. [27] perform hard negative mining of training patches in a way that simulates the problem of self-similarity when matching at the image level. Revaud et al. [39] train a detector to find few but reliable key points. Similarly Cieslewski et al. [12] learn to find key points with high probability of being inliers in robust model fitting.

In this work, we take a more radical approach. Instead of hand-crafting a training procedure that emulates aspects of high-level vision pipelines, we embed the feature detector in a complete vision pipeline during training. Particularly, our pipeline addresses the task of relative pose estimation, a central component in camera re-localization, structure-from-motion or SLAM. The pipeline incorporates key point selection, descriptor matching and robust model fitting. We do not need to pre-define ground truth correspondences, dispensing with the need for hard-negative mining. Furthermore, we do not need to speculate whether it is more beneficial to find many matches or few, reliable matches. All these aspects are solely guided by the task loss, i.e., by minimizing the relative pose error between two images.

Key point selection and descriptor matching are discrete operations which cannot be directly differentiated. However, since many feature detectors predict key point locations as heat maps, we can reformulate key point selection as a sampling operation. Similarly, we lift feature matching to a distribution where the probability of a match stems from its descriptor distance. This allows us to apply principles from reinforcement learning [48] to directly optimize a high-level task loss. Particularly, all operations after the feature matching stage, e.g., robust model fitting, do not need to be differentiable since they only provide a reward signal for learning. In summary, our training methodology puts little restrictions on the feature detection architecture or the vision task to be optimized for.

We demonstrate our approach using the SuperPoint detector [14], which regularly ranks among top methods in independent evaluations [16, 5, 39]. We train SuperPoint for the task of relative pose estimation by robust fitting of the essential matrix. For this task, our training procedure closes the gap between SuperPoint and a state-of-the-art SIFT-based pipeline, see Fig. 1 for a comparison of results.

We summarize our main contributions:

- A new training methodology which allows for learning a feature detector and descriptor, embedded in a complete vision pipeline, to optimize its performance for a high-level vision task.
- We apply our method to a state-of-the-art architecture, Superpoint [14], and train it for the task of relative pose estimation.
- After training, SuperPoint [14] reaches, and slightly exceeds, the accuracy of SIFT [26] which previously achieved best results for this task.

2. Related Work

Of all hand-crafted feature detectors, SIFT [26] stands out for its long lasting success. SIFT finds key point locations as a difference-of-Gaussian filter response in the scale space of an image, and describes features using histograms of oriented gradients [13]. Arandjelovic and Zisserman [1] improve the matching accuracy of SIFT by normalizing its descriptor, also called RootSIFT. Other hand-crafted feature detectors improve efficiency for real-time applications while sacrificing as little accuracy as possible [6, 40].

MatchNet [18] is an early example of learning to compare image patches using a patch similarity network. The reliance on a network as a similarity measure prevents the use of efficient nearest neighbor search schemes. L2-Net [50], and subsequent works, instead learn patch descriptors to be compared using the Euclidean distance. Balntas et al. [54] demonstrated the advantage of using a triplet loss for descriptor learning over losses defined on pairs of patches only. A triplet combines two matching and one non-matching patch, and the triplet loss optimizes relative distances within a triplet. HardNet [29] employs a “hardest-in-batch” strategy when assembling triplets for training, i.e., for each matching patch pair, they search for the most similar non-matching patch within a mini-batch. GeoDesc [27] constructs mini-batches for training that contain visually similar but non-matching patch pairs to mimic the problem of self-similarity when matching two images. SOSNet [51] uses second order similarity regularization to enforce a structure of the descriptor space that leads to well separated clusters of similar patches.

Learning feature detection has also started to attract attention recently. ELF [7] shows that feature detection can be implemented using gradient tracing within a pre-trained neural network. Key.Net [5] combines hand-crafted and learned filters to avoid overfitting. The detector is trained using a repeatability objective, i.e., finding the same points
in two related images, synthetically created by homography warping. SLPs [12] learns to predict a pixel-wise probability map of inlier locations as key points, inlier being a correspondence which can be continuously tracked throughout an image sequence by an off-the-shelf feature tracker.

LIFT [58] was the first, complete learning-based architecture for feature detection and description. It rebuilds the main processing steps of SIFT with neural networks, and is trained using sets of matching and non-matching image patches extracted from structure-from-motion datasets. DELF [35] learns detection and description for image retrieval, where coarse key point locations emerge by training an attention layer on top of a dense descriptor tensor. D2-Net [16] implements feature detection and description by searching for local maxima in the filter response map of a pre-trained CNN. R2D2 [39] proposes a learning scheme for identifying feature locations that can be matched uniquely among images, avoiding repetitive patterns.

All mentioned learning-based works design training schemes that emulate difficult conditions for a feature detector when employed for a vision task. Our work is the first to directly embed feature detection and description in a complete vision pipeline for training where all real-world challenges occur, naturally. On a similar note, Keypoint-sorter when employed for a vision task. Our work is the first, complete learning-based architecture for feature detection and description. It rebuilds the main processing steps of SIFT with neural networks, and is trained using sets of matching and non-matching image patches extracted from structure-from-motion datasets. DELF [35] learns detection and description for image retrieval, where coarse key point locations emerge by training an attention layer on top of a dense descriptor tensor. D2-Net [16] implements feature detection and description by searching for local maxima in the filter response map of a pre-trained CNN. R2D2 [39] proposes a learning scheme for identifying feature locations that can be matched uniquely among images, avoiding repetitive patterns.

We observe how well these key points and their matching perform in the vision task, and adjust network parameters \( w \) such that an outcome with low loss becomes more probable. We show an overview of our approach in Fig. 2.

In the following, we firstly describe how to reformulate key point selection and feature matching as probabilistic actions. Thereafter, we formulate our learning objective, and how to efficiently approximate it using sampling.

3.1. Probabilistic Key Point Selection

We assume that the detection network predicts a key point heat map \( f(I; w) \) for an input image, as is common in many architectures [58, 14, 39, 12]. Feature locations are usually selected from \( f(I; w) \) by taking all local maxima combined with local non-max suppression.

To make key point selection probabilistic, we instead interpret the heat map as a probability distribution over key point locations \( f(I; w) = P(x; w) \) parameterized by the network parameters \( w \). We define a set of \( N \) key points for image \( I \) as \( \mathcal{X} = \{x_i\} \) sampled independently according to

\[
P(\mathcal{X}; w) = \prod_{i=1}^{N} P(x_i; w),
\]

apply a robust estimator like RANSAC [17] with a 5-point solver [33] to find the essential matrix which maximises the inlier count among all correspondences. An inlier is defined as a correspondence with a distance to the closest epipolar line below a threshold [20]. Decomposition of the essential matrix yields an estimate of the relative transformation \( T \).

We implement feature detection using two networks: a detection network and a description network. In practice, we use a joint architecture, SuperPoint[14], where most weights are shared between detection and description. The main goal of this work is to optimize the learnable parameters \( w \) of both networks such that their accuracy for the vision task is enhanced. For our application, the networks should predict key points and descriptors such that the relative pose error between two images is minimized. Key point selection and feature matching are discrete, non-differentiable operations. Therefore, we cannot directly propagate gradients of our estimated transformation \( T \) back to update the network weights, as in standard supervised learning. Components of our vision pipeline, like the robust estimator (e.g., RANSAC [17]) or the minimal solver (e.g., the 5-point solver [33]) might also be non-differentiable. To optimize the neural network parameters for our task, we apply principles from reinforcement learning [48]. We formulate feature detection and matching as probabilistic actions where the probability of taking an action, i.e., selecting a key point, or matching two features, depends on the output of the neural networks. During training, we sample different instantiations of key points and their matchings based on the probability distributions predicted by the neural networks.
Relative Pose Estimation
1) probabilistic key point selection 2) probabilistic matching 3) high-level vision task (non-differentiable black box)

Figure 2. Method. Top: Our training pipeline consists of probabilistic key point selection and probabilistic feature matching. Based on established feature matches, we solve for the relative pose and compare it to the ground truth pose. We treat the vision task as a (potentially non-differentiable) black box. It provides an error signal, used to reinforce the key point and matching probabilities. Based on the error signal both CNNs (green and blue) are updated, making a low loss more likely. Bottom Left: We sample key points according to the heat map predicted by the detector. Bottom Right: We implement probabilistic matching by, firstly, doing an exhaustive matching between all key points independently in each image as

\[ P(X^i, X'; w) = P(X^i; w)P(X'; w). \tag{2} \]

3.2. Probabilistic Feature Matching

We assume that a second description network predicts a feature descriptor \( d(x; w) \) for a given key point \( x \). To simplify notation, we use \( w \) to denote the learnable parameters associated with feature detection and description. We define a feature match as a pairing of one key point from image \( I \) and image \( I' \), respectively; \( m_{ij} = (x_i, x'_j) \). We give the probability of a match between two key points \( x_i \) and \( x'_j \) as a function of their descriptor distance,

\[ P(m_{ij}|X^i, X'; w) = \frac{\exp[-||d(x_i; w) - d(x'_j; w)||]}{\sum_{m_{k,l}} \exp[-||d(x_k; w) - d(x'_l; w)||]]. \tag{3} \]

Note that the matching probability is conditioned on the sets of key points which we selected in an earlier step. The matching distribution is normalized using all possible matches \( m_{k,l} = (x_k, x'_l) \) with \( x_k \in X^i \) and \( x'_l \in X^{i'} \). The matching distribution assigns a low probability to a match, if the associated key points have very different descriptors. In turn, if the network wants to increase the probability of a (good) match during training, it has to reduce the descriptor distance of the associated key points relative to all other matches for the same image pair.

We define a complete set of \( M \) matches \( M = \{m_{ij}\} \) between \( I \) and \( I' \) sampled independently according to

\[ P(M|X^i, X^{i'}; w) = \prod_{m_{ij} \in M} P(m_{ij}|X^i, X^{i'}; w). \tag{4} \]

3.3. Learning Objective

We learn network parameters \( w \) in a supervised fashion, i.e., we assume to have training data of the form \((I, I', T^*)\) with ground truth transformation \( T^* \). Note that we do not need ground truth key point locations \( X \) or ground truth image correspondences \( M \).

Our learning formulation is agnostic to the implementation details of how the vision task is solved using tentative image correspondences \( M \). We treat the exact processing pipeline after the feature matching stage as a black box which produces only one output: a loss value \( \ell(M, X^i, X^{i'}) \), which depends on the key points \( X \) and \( X' \) that we selected for both images, and the matches \( M \) that we selected among the key points. For relative pose estimation, calculating \( \ell \) entails robust fitting of the essential matrix, its decomposition to yield an estimated relative camera transformation \( \hat{T} \), and its comparison to a ground truth transformation \( T^* \). We only require the loss value itself, not its gradients.

Our training objective aims at reducing the expected task loss when sampling key points and matches according to the probability distributions parameterized by the learnable
parameters \( w \):

\[
L(\mathbf{w}) = \mathbb{E}_{\mathcal{M}, \mathcal{X}, \mathcal{X}' \sim P(\mathcal{M}, \mathcal{X}, \mathcal{X}'; \mathbf{w})} \left[ \ell(\mathcal{M}, \mathcal{X}, \mathcal{X}') \right]
= \mathbb{E}_{\mathcal{X}, \mathcal{X}' \sim P(\mathcal{X}, \mathcal{X}'; \mathbf{w})} \mathbb{E}_{\mathcal{M} \sim P(\mathcal{M}|\mathcal{X}, \mathcal{X}'; \mathbf{w})} \left[ \mathbb{E}_{\mathcal{M}|\mathcal{X}, \mathcal{X}'} \left[ \ell(\cdot) \right] \right],
\]

where we abbreviate \( \ell(\mathcal{M}, \mathcal{X}, \mathcal{X}') \) to \( \ell(\cdot) \). We split the expectation in key point selection and match selection. Firstly, we select key points \( \mathcal{X} \) and \( \mathcal{X}' \) according to the heat map predictions of the detection network \( P(\mathcal{X}, \mathcal{X}'; \mathbf{w}) \) (see Eq. 1 and 2). Secondly, we select matches among these key points according to a probability distribution \( P(\mathcal{M}|\mathcal{X}, \mathcal{X}'; \mathbf{w}) \) calculated from descriptor distances (see Eq. 3 and 4).

Calculating the expectation and its gradients exactly would necessitate summing over all possible key point sets, and all possible matchings, which is clearly infeasible. To make the calculation tractable, we assume that the network is already initialized, and makes sensible predictions that we aim at optimizing further for our task. In practice, we take an off-the-shelf architecture, like SuperPoint [14], which was trained on a low-level matching task. For such an initialized network, we observe the following properties:

1. Heat maps predicted by the feature detector are sparse. The probability of selecting a key point is zero at almost all image pixels (see Fig. 2 bottom, left). Therefore, only few image locations have an impact on the expectation.
2. Matches among unrelated key points have a large descriptor distance. Such matches have a probability close to zero, and no impact on the expectation.

Observation 1) means, we can just sample from the key point heat map, and ignore other image locations. Observation 2) means that for the key points we selected, we do not have to realise a complete matching of all key points in \( \mathcal{X} \) to all key points in \( \mathcal{X}' \). Instead, we rely on a k-nearest-neighbour matching with some small \( k \). All nearest neighbours beyond \( k \) likely have large descriptor distances, and hence near zero probability. In practice, we found no advantage in using a \( k > 1 \) which means we can do a normal nearest neighbour matching during training when calculating \( P(\mathcal{M}|\mathcal{X}, \mathcal{X}'; \mathbf{w}) \) (see Fig. 2 bottom, right).

We update the learnable parameters \( \mathbf{w} \) according to the gradients of Eq. 5, following the classic REINFORCE algorithm [55] of Williams:

\[
\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}) = \mathbb{E}_{\mathcal{X}, \mathcal{X}'} \left[ \mathbb{E}_{\mathcal{M}|\mathcal{X}, \mathcal{X}'} \left[ \ell(\cdot) \right] \frac{\partial}{\partial \mathbf{w}} \log P(\mathcal{X}, \mathcal{X}'; \mathbf{w}) \right]
+ \mathbb{E}_{\mathcal{X}, \mathcal{X}'} \left[ \mathbb{E}_{\mathcal{M}|\mathcal{X}, \mathcal{X}'} \left[ \ell(\cdot) \frac{\partial}{\partial \mathbf{w}} \log P(\mathcal{M}|\mathcal{X}, \mathcal{X}'; \mathbf{w}) \right] \right].
\]

Note that we only need to calculate the gradients of the log probabilities of key point selection and feature matching. We approximate the expectations in the gradient calculation by sampling. We approximate \( \mathbb{E}_{\mathcal{X}, \mathcal{X}'} \) by drawing \( n_X \) samples \( \hat{\mathcal{X}}, \hat{\mathcal{X}}' \sim P(\mathcal{X}, \mathcal{X}'; \mathbf{w}) \). For a given key point sample, we approximate \( \mathbb{E}_{\mathcal{M}|\mathcal{X}, \mathcal{X}'} \) by drawing \( n_M \) samples \( \hat{\mathcal{M}} \sim P(M|\mathcal{X}, \mathcal{X}'; \mathbf{w}) \). For each sample combination, we run the vision pipeline and observe the associated task loss \( \ell \). To reduce the variance of the gradient approximation, we subtract the mean loss over all samples as a baseline [48]. We found a small number of samples for \( n_X \) and \( n_M \) sufficient for the pipeline to converge.

4. Experiments

We train the SuperPoint [14] architecture for the task of relative pose estimation, and report our main results in Sec. 4.1. Furthermore, we analyse the impact of reinforcing SuperPoint for relative pose estimation on a low-level matching benchmark (Sec. 4.2), and in a structure-from-motion task (Sec. 4.3).

4.1. Relative Pose Estimation

Network Architecture. SuperPoint [14] is a fully-convolutional neural network which processes full-sized images. The network has two output heads: one produces a heat map from which key points can be picked, and the other head produces 256-dimensional descriptors as a dense descriptor field over the image. The descriptor output of SuperPoint fits well into our training methodology, as we can look up descriptors for arbitrary image locations without doing repeated forward passes of the network. Both output heads share a common encoder which processes the image and reduces its dimensionality, while the output heads act as decoders. We use the network weights provided by the authors as an initialization.

Task Description. We calculate the relative camera pose between a pair of images by robust fitting of the essential matrix. We show an overview of the processing pipeline in Fig. 2. The feature detector produces a set of tentative image correspondences. We estimate the essential matrix using the 5-point algorithm [33] in conjunction with a robust estimator. For the robust estimator, we conducted experiments with a standard RANSAC [17] estimator, as well as with the recent NG-RANSAC [10]. NG-RANSAC uses a neural network to suppress outlier correspondences, and to guide RANSAC sampling towards promising candidates for the essential matrix. As a learning-based robust estimator, NG-RANSAC is particularly interesting in our setup, since we can refine it in conjunction with SuperPoint during end-to-end training.

Datasets. To facilitate comparison to other methods, we follow the evaluation protocol of Yi et al. [59] for relative pose estimation. They evaluate using a collection of 7 outdoor and 16 indoor datasets from various sources
We apply a square root soft clamping \( \left[ \right] \) to limit the influence of such large losses, \( \text{Eq. 5} \). To approximate the expected task loss \( C(w) \) and its gradients in \text{Eq. 5} and \text{Eq. 6}, we draw key points \( n_X = 3 \) times, and, for each set of key points, we draw \( n_M = 3 \) sets of matches. Therefore, for each training iteration, we run the vision pipeline 9 times, which takes 1.5s to 2.1s on a single Tesla K80 GPU, depending on the termination of the robust estimator. We train using the Adam \[23\] optimizer and a learning rate of \( 10^{-7} \) for 150k iterations which takes approximately 60 hours. Our training code is based on PyTorch \[37\] for SuperPoint \[14\] integration and learning, and on OpenCV \[11\] for estimating the relative pose. We will make our source code publicly available to ensure reproducibility of our approach.

For testing, we revert to a deterministic procedure for feature detection, instead of doing sampling. We select the strongest 2000 key points from the detector heat map using local non-max suppression. We remove very weak key points with a heat map value below 0.00015. We do a nearest neighbor matching of the corresponding feature descriptors, and keep all matches of mutual nearest neighbors. We adhere to this procedure for SuperPoint before and after our proposed training (denoted Reinforced SP).

Firstly, we train and test our pipeline using a standard RANSAC estimator for essential matrix fitting, see \text{Fig. 3 a}). We compare to a state-of-the-art SIFT-based \[26\] pipeline, which uses RootSIFT descriptor normalization \[1\]. For RootSIFT, we apply Lowe’s ratio criterion \[26\] to filter matches where the distance ratio of the nearest and second nearest neighbor is above 0.8. We also compare to the LIFT feature detector \[58\], with and without the learned inlier classification scheme of Yi et al. \[59\] (denoted InClass). Finally, we compare the results of SuperPoint \[14\] before and after our proposed training (denoted Reinforced SP).

Reinforced SuperPoint exceeds the accuracy of SuperPoint across all thresholds, proving that our training scheme indeed optimizes the performance of SuperPoint for relative pose estimation. The effect is particularly strong for outdoor environments. For indoors, the training effect is weaker, because large texture-less areas make these scenes difficult for sparse feature detection, in principle. SuperPoint exceeds the accuracy of LIFT by a large extent, but does not reach the accuracy of RootSIFT. We found that the excellent accuracy of RootSIFT is largely due to the effectiveness of Lowe’s ratio filter for removing unreliable SIFT matches. We tried the ratio filter also for SuperPoint, but we found no ratio threshold value that would consistently improve accuracy across all datasets.

To implement a similarly effective outlier filter for SuperPoint, we substitute the RANSAC estimator in our vision pipeline with the recent learning-based NG-RANSAC \[10\] estimator. We train NG-RANSAC for SuperPoint using the public code of Brachmann and Rother \[10\], and with the initial weights for SuperPoint by Detone et al. \[14\]. With NG-RANSAC as a robust estimator, SuperPoint almost reaches the accuracy of RootSIFT, see \text{Fig. 3, b}). Finally, we embed both, SuperPoint and NG-RANSAC in our vision pipeline, and train them jointly and end-to-end. After our training schema, Reinforced SuperPoint matches and slightly exceeds the accuracy of RootSIFT. \text{Fig. 3, c)} shows an ablation study where we either update only NG-RANSAC, only SuperPoint or both during end-to-end training. While the main improvement comes from updating SuperPoint, updating NG-RANSAC as well allows the robust estimator to adapt to the changing matching statistics of SuperPoint throughout the training process.

### Analysis.

We visualize the effect of our training procedure on the outputs of SuperPoint in \text{Fig. 4}. For the key point heat maps, we observe two major effects. Firstly, many key points seem to be discarded, especially for repetitive patterns that would result in ambiguous matches. Secondly, some key points are kept, but their position is adjusted, presumably to achieve a lower relative pose error. For the descriptor distribution, we see a tendency of reducing the descriptor distance for correct matches, and increas-
Figure 3. Relative Pose Estimation. a) AUC of the relative pose error using a RANSAC estimator for the essential matrix. Results of RootSIFT as reported in [10], results of LIFT as reported in [59]. b) AUC using NG-RANSAC [10] as robust estimator. c) For our best result, we show the impact of training SuperPoint vs. NG-RANSAC end-to-end. Init. for SuperPoint means weights provided by Detone et al. [14], init. for NG-RANSAC means training according to Brachmann and Rother [10] for SuperPoint. We show results worse than the RootSIFT baseline in red, and results better than or equal to RootSIFT in green.

Figure 4. Effect of Training. a) We visualize the difference in key point heat maps predicted by SuperPoint before and after our end-to-end training. Key points which appear blue were discarded, key points with a gradient from blue to red were moved. b) We create a fixed set of matches using (initial) SuperPoint, and visualize the difference in matching probabilities for these matches before and after our end-to-end training. The probability of red matches increased by reducing their descriptor distance relative to all other matches.

4.2. Low-Level Matching Accuracy

We investigate the effect of our training scheme on low-level matching scores. Therefore, we analyse the performance of Reinforced SuperPoint, trained for relative pose estimation (see previous section), on the H-Patches [3] benchmark. The benchmark consists of 116 test sequences showing images under increasing viewpoint and illumination changes. We adhere to the evaluation protocol of Dusmanu et al. [16]. That is, we find key points and matches between image pairs of a sequence, accepting only matches of mutual nearest neighbours between two images. We calculate the reprojection error of each match using the ground truth homography. We measure the average percentage of correct matches for thresholds ranging from 1px to 10px for the reprojection error. We compare to a RootSIFT [11] baseline with a hessian affine detector [28] (denoted HA+RootSIFT) and several learned detectors, namely HardNet++ [29] with learned affine normalization [30] (denoted HANet+HN++), LF-Net [36], DELF [35] and D2-Net [16]. The original SuperPoint [14] beats all competitors in terms of AUC when combining illumination and viewpoint sequences. In particular, SuperPoint significantly exceeds the matching accuracy of RootSIFT on H-Patches, although RootSIFT outperforms SuperPoint in the task of relative pose estimation. This confirms that low-level matching accuracy does not necessarily translate to accuracy in a high-level vision task, see our earlier discussion. As for Reinforced SuperPoint, we observe an increased matching accuracy compared to SuperPoint, due to having fewer but more reliable and precise key points.

4.3. Structure-from-Motion

We evaluate the performance of Reinforced SuperPoint, trained for relative pose estimation, in a structure-from-
Figure 5. Evaluation on H-Patches [3]. Left: We show the mean matching accuracy for SuperPoint before and after being trained for relative pose estimation. Results of competitors as reported in [16]. Right: Area under the curve (AUC) for the plots on the left.

### Outdoors

<table>
<thead>
<tr>
<th>Method</th>
<th>Kps</th>
<th>Matches</th>
<th>Inliers</th>
<th>GT Inl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperPoint (SP)</td>
<td>1993</td>
<td>1008</td>
<td>24.8%</td>
<td>21.9%</td>
</tr>
<tr>
<td>Reinf. SP (ours)</td>
<td>1892</td>
<td>955</td>
<td>28.4%</td>
<td>25.3%</td>
</tr>
</tbody>
</table>

### Indoors

<table>
<thead>
<tr>
<th>Method</th>
<th>Kps</th>
<th>Matches</th>
<th>Inliers</th>
<th>GT Inl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperPoint (SP)</td>
<td>1247</td>
<td>603</td>
<td>13.4%</td>
<td>9.6%</td>
</tr>
<tr>
<td>Reinf. SP (ours)</td>
<td>520</td>
<td>262</td>
<td>16.4%</td>
<td>11.1%</td>
</tr>
</tbody>
</table>

Table 1. Average number of key points and matches found by SuperPoint before and after our training. We also report the estimated ratio of inliers, and the ground truth ratio of inliers.

<table>
<thead>
<tr>
<th>Dataset (11 img.)</th>
<th>Method</th>
<th># Sparse Points</th>
<th>Track Len.</th>
<th>Repr. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fountain</td>
<td>DSP-SIFT</td>
<td>15k</td>
<td>4.79</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>GeoDesc</td>
<td>17k</td>
<td>4.99</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>SuperPoint</td>
<td>31k</td>
<td>4.75</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>Reinf. SP</td>
<td>9k</td>
<td>4.86</td>
<td>0.87</td>
</tr>
<tr>
<td>Herzjesu (8 img.)</td>
<td>DSP-SIFT</td>
<td>8k</td>
<td>4.22</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>GeoDesc</td>
<td>9k</td>
<td>4.34</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>SuperPoint</td>
<td>21k</td>
<td>4.10</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>Reinf. SP</td>
<td>7k</td>
<td>4.32</td>
<td>0.82</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset (128 img.)</th>
<th>Method</th>
<th># Sparse Points</th>
<th>Track Len.</th>
<th>Repr. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>South Building</td>
<td>DSP-SIFT</td>
<td>113k</td>
<td>5.92</td>
<td>0.58</td>
</tr>
<tr>
<td></td>
<td>GeoDesc</td>
<td>170k</td>
<td>5.21</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>SuperPoint</td>
<td>160k</td>
<td>7.83</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td>Reinf. SP</td>
<td>102k</td>
<td>7.86</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 2. Effect of our end-to-end training on a structure-from-motion benchmark. Reinf. SP denotes SuperPoint after being trained for relative pose estimation. Reprojection error is in px.

We have presented a new methodology for end-to-end training of feature detection and description which includes key point selection, feature matching and robust model estimation. We applied our approach to the task of relative pose estimation between two images. We observe that our end-to-end training increases the pose estimation accuracy of a state-of-the-art feature detector by removing unreliable key points, and refining the locations of remaining key points. We require a good initialization of the network, which might have a limiting effect in training. In particular, we observe that the network rarely discovers new key points. Key point locations with very low initial probability will never be selected, and cannot be reinforced. In future work, we could combine our training schema with importance sampling, for biased sampling of interesting locations.

### Acknowledgements

This project has received funding from the European Social Fund (ESF) and Free State of Saxony under SePIA grant 100299506, DFG Cluster of Excellence CeTI (EXC2050/1 Project ID 390696704), DFG grant 389792660 as part of TRR 248, the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 647769), and DFG grant COVMAP: Intelligente Karten mittels gemeinsamer GPS- und Videodatenanalyse (RO 4804/2-1). The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.
References